

SELENIUM

Автоматизированное тестирование Web сайтов

Практическое пособие по автоматизированному тестированию.

Содержание

<u>Введение</u>	-----	2
<u>Тест-кейс</u>	-----	4
<u>Установка и запуск Selenium Grid</u>	-----	5
<u>Создание проекта Selenium (JavaScript) с простым автотестом</u>	-----	7
<u>Создание проекта Selenium (Java) с простым автотестом</u>	-----	11
<u>Практика применения JUnit</u>	-----	15
<u>Описание паттернов PageObjects и StepObjects</u>	-----	19
<u>Запуск автотестов с помощью Maven</u>	-----	21
<u>Практика применения TestNG</u>	-----	23
<u>Создание проекта Selenium (Python) с простым автотестом</u>	-----	26
<u>Практика применения Unittest</u>	-----	31
<u>Практика применения PyTest</u>	-----	33
<u>Практика PyTest в стиле xUnit</u>	-----	38
<u>Практика PyTest с применением Fixture</u>	-----	40
Фреймворк WebdriverIO	-----	
Фреймворк Cucumber	-----	
Фреймворк Selenide	-----	
Фреймворк Robot	-----	
<u>Фреймворк Codeception</u>	-----	
<u>Отчет Allure Report</u>	-----	
<u>Практика применения Allure Report (Java)</u>	-----	
<u>Практика применения Allure Report PyTest (Python)</u>	-----	
Отчет Report Portal	-----	
Jenkins	-----	
Docker	-----	
Нагрузочное тестирование с помощью Jmeter	-----	
<u>Тестирование API с помощью Bruno</u>	-----	
<u>Локаторы XPATH и сравнение с CSS</u>	-----	

Введение

В данной книге описана практика разработки автоматизированных тестов на основе технологии Selenium с использованием специальных фреймворков: Cucumber, Selenide, Robot, WebdriverIO, Codeception.

Для создания автотестов будут применятся технологии модульного тестирования JUnit, TestNG, Unitest, PyTest, PHPUnit.

В демонстрационных примерах отражена полноценная разработка автотестов через паттерны PageObject, StepsObject.

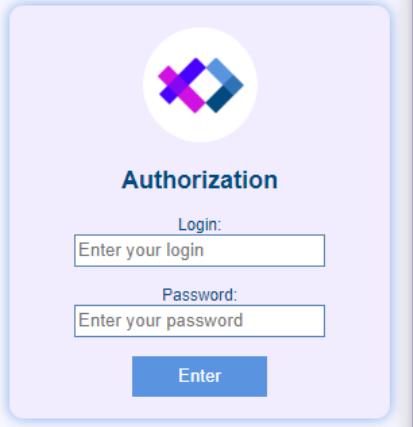
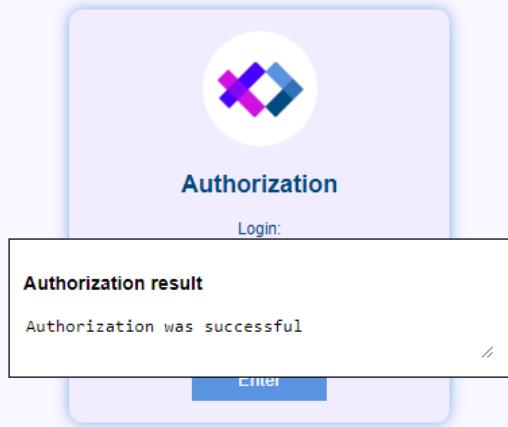
В дополнение практических занятий будет рассмотрено нагружочное тестирование (Jmeter), тестирование API функций (Bruno).

Полезные ссылки

- Официальный сайт Selenium:
<https://www.selenium.dev/>
- Документация Selenium
<https://www.selenium.dev/documentation/overview/>
- Быстрый старт Selenium
https://www.selenium.dev/documentation/grid/getting_started/
- Скачать Selenium Server (Grid):
<https://www.selenium.dev/downloads/>
- Официальная страница Chrome драйвер
<https://googlechromelabs.github.io/chrome-for-testing/#stable>
- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Официальная страница Visual Studio Code
<https://code.visualstudio.com/>
- Официальная страница NodeJS
<https://nodejs.org/>
- Официальная страница IntelliJ IDEA Community Edition
<https://www.jetbrains.com/idea/download/other.html>
- Официальная страница Maven
<https://mvnrepository.com/>
- Официальная страница junit5
<https://junit.org/junit5/>
- Официальная страница TestNG
<https://testng.org/>
- Официальная страница Java SE Development Kit 11
<https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>
- Официальная страница PyCharm Community Edition
<https://www.jetbrains.com/pycharm/download/other.html>
- Официальная страница Python
<https://www.python.org/>

Тест-кейс

Во всех примерах данной книги будет использоваться тест-кейс на проверку авторизации.

Suit ID: (id набора тестов)	S1	
Case ID: (id тест-кейса)	01	
Priority: (приоритет)	Minor / Medium / Major / Минорно / Средне / Важно	
Summary: (заголовок)	Проверка авторизации на сайте	
Environment: (окружение)	Технология автоматизированного тестирования Selenium	
Requirement (требования)	Установить следующее программное обеспечение: Selenium, ChromeDriver, Java, Python, NodeJS.	
Created (создано)	Date: 01.01.25 (дата создания)	Author: SomovQA (автор)
Data: (данные)	тестовая страница https://somovstudio.github.io/test_eng.html корректный тестовый логин: admin корректный тестовый пароль: 0000	
Attachments (вложения)	 	
Steps: (шаги)	<ol style="list-style-type: none"> Открыть страницу авторизации В поле Login ввести корректное имя пользователя В поле Password ввести корректный пароль пользователя Нажать кнопку Enter 	
Expected result: (ожидаемый результат)	Должен открыться поп-ап с сообщением "Authorization was successful"	
Actual result: (фактический результат)	Автотест вернул успешный результат проверки	
Status: (статус)	passed / failed	
Executed: (выполнено)	Date: 01.01.25 (дата выполнения)	By: Selenium (исполнитель)

Установка и запуск Selenium Grid

Чтобы настроить Selenium Server нужно выполнить следующие действия.

Скачать и установить [Java SE Development Kit 11](#)

Скачать и установить браузер [Chrome](#)

Скачать драйвер [ChromeDriver](#)

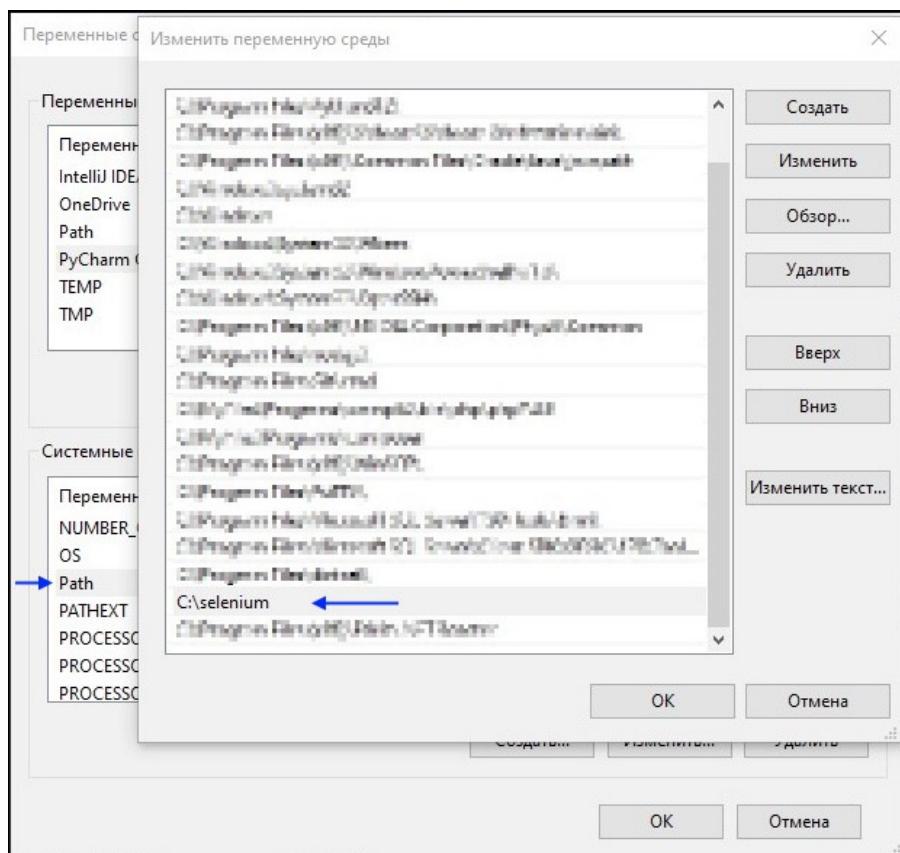
Скачать jar файл [Selenium Server](#) или с [github](#)

Создать папку C:\selenium\ в которую скопировать файлы:

selenium-server-4.24.0.jar, chromedriver.exe

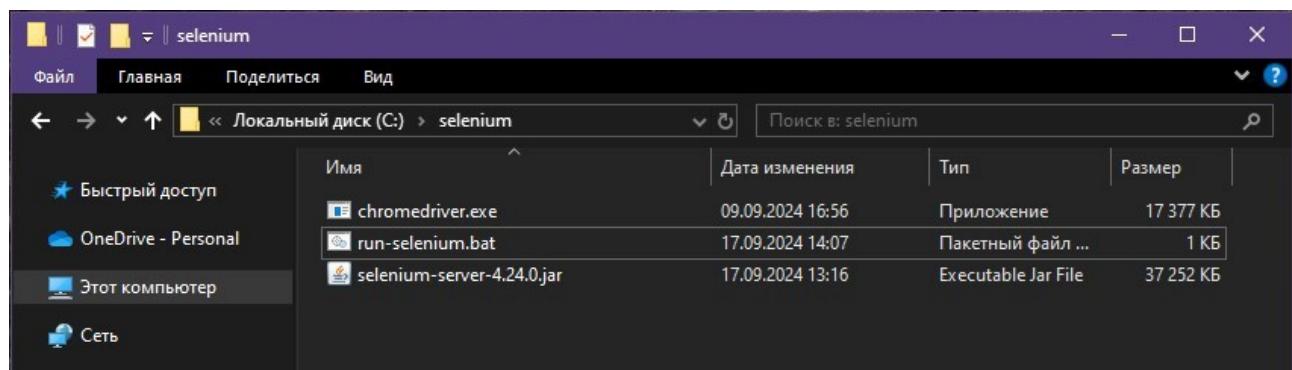
В переменной Path прописать путь к папке C:\selenium

(Панель управления > Система > Дополнительные параметры системы > Переменные среды)

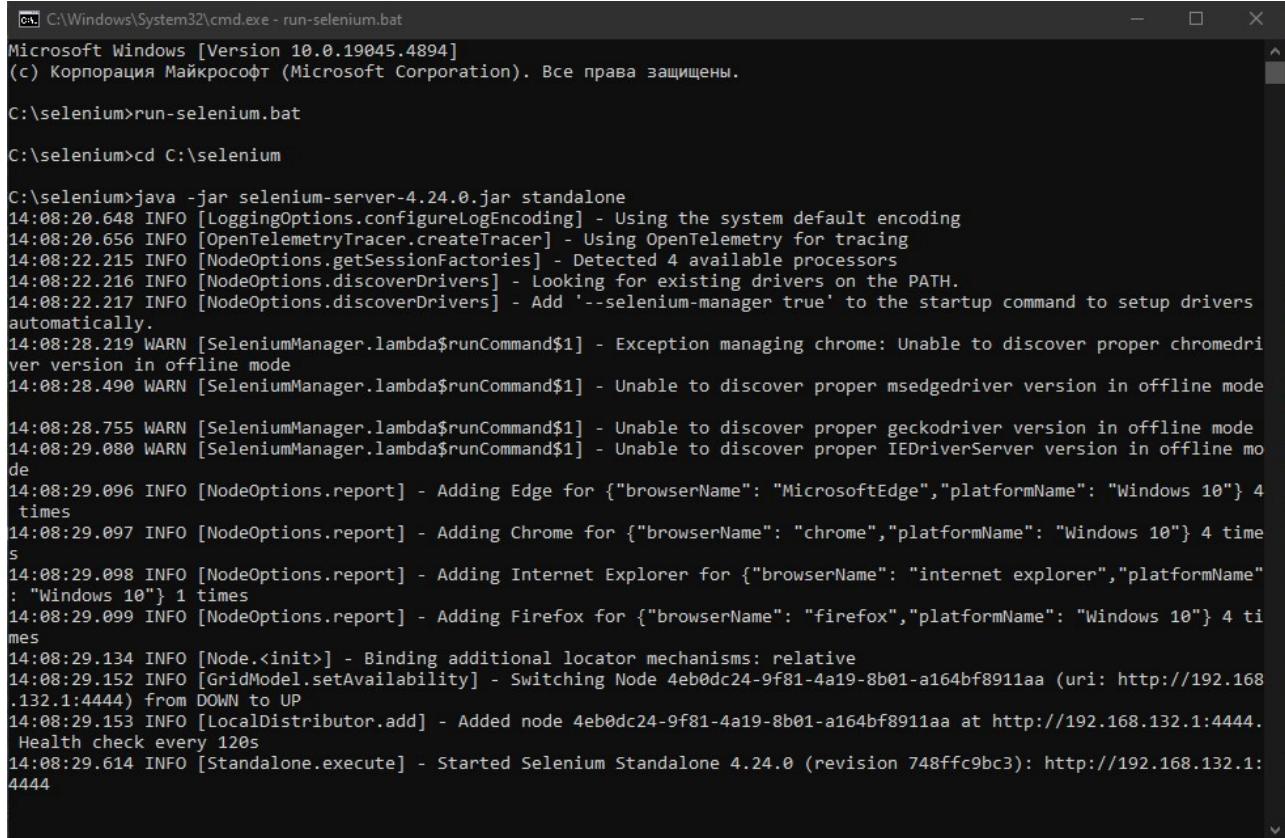


В папке C:\selenium\ создать файл run-selenium.bat в котором написать:

```
cd C:\selenium
java -jar selenium-server-4.24.0.jar standalone
```



Запустить файл run-selenium.bat



```
C:\Windows\System32\cmd.exe - run-selenium.bat
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\selenium>run-selenium.bat
C:\selenium>cd C:\selenium

C:\selenium>java -jar selenium-server-4.24.0.jar standalone
14:08:20.648 INFO [LoggingOptions.configureLogEncoding] - Using the system default encoding
14:08:20.656 INFO [OpenTelemetryTracer.createTracer] - Using OpenTelemetry for tracing
14:08:22.215 INFO [NodeOptions.getSessionFactories] - Detected 4 available processors
14:08:22.216 INFO [NodeOptions.discoverDrivers] - Looking for existing drivers on the PATH.
14:08:22.217 INFO [NodeOptions.discoverDrivers] - Add '--selenium-manager true' to the startup command to setup drivers automatically.
14:08:28.219 WARN [SeleniumManager.lambda$runCommand$1] - Exception managing chrome: Unable to discover proper chromedriver version in offline mode
14:08:28.490 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper msedgedriver version in offline mode

14:08:28.755 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper geckodriver version in offline mode
14:08:29.088 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper IEDriverServer version in offline mode
14:08:29.096 INFO [NodeOptions.report] - Adding Edge for {"browserName": "MicrosoftEdge", "platformName": "Windows 10"} 4 times
14:08:29.097 INFO [NodeOptions.report] - Adding Chrome for {"browserName": "chrome", "platformName": "Windows 10"} 4 times
14:08:29.098 INFO [NodeOptions.report] - Adding Internet Explorer for {"browserName": "internet explorer", "platformName": "Windows 10"} 1 times
14:08:29.099 INFO [NodeOptions.report] - Adding Firefox for {"browserName": "firefox", "platformName": "Windows 10"} 4 times
14:08:29.134 INFO [Node.<init>] - Binding additional locator mechanisms: relative
14:08:29.152 INFO [GridModel.setAvailability] - Switching Node 4eb0dc24-9f81-4a19-8b01-a164bf8911aa (uri: http://192.168.132.1:4444) from DOWN to UP
14:08:29.153 INFO [LocalDistributor.add] - Added node 4eb0dc24-9f81-4a19-8b01-a164bf8911aa at http://192.168.132.1:4444. Health check every 120s
14:08:29.614 INFO [Standalone.execute] - Started Selenium Standalone 4.24.0 (revision 748ffc9bc3): http://192.168.132.1:4444
```

Чтобы остановить Selenium нажмите в консоли Ctrl+C и затем Y

Создание проекта Selenium (JavaScript) с простым автотестом

Для разработки автотестов с помощью JavaScript нужно выполнить следующие действия.

Скачать и установить [Visual Studio Code](#)

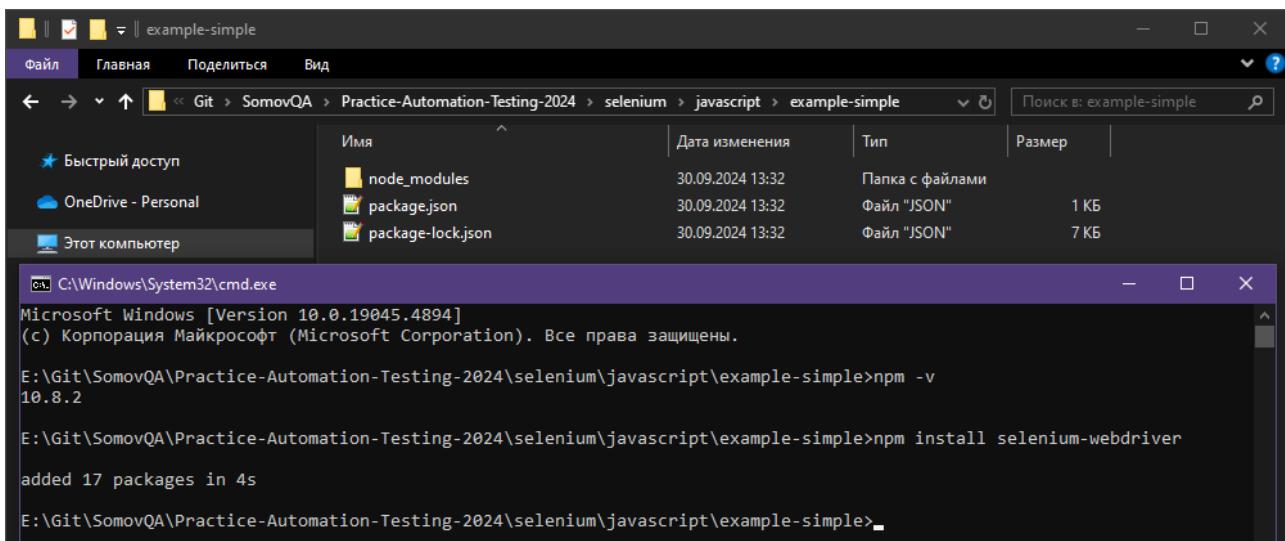
Скачать и установить [NodeJS](#)

Открыть консоль и проверить работу NodeJS с помощью команды:

```
npm -v
```

Создать папку проекта example-simple и в корне этой папки выполнить в консоли команду:

```
npm install selenium-webdriver
```



Открыть папку в редакторе Visual Studio Code

Создать файл test.js и описать автотест следующим образом

```
(async function example() {
    let driver = await new Builder().forBrowser(Browser.CHROME).build();
    try {
        await driver.get('https://somovstudio.github.io/test_eng.html');
        await driver.findElement(By.name('login')).sendKeys('admin');
        await driver.findElement(By.name('pass')).sendKeys('0000');
        await driver.findElement(By.id('buttonLogin')).click();
        let element = await driver.findElement(By.id('result'));
        await driver.wait(until.elementIsVisible(element), 5000);
        let text = await driver.findElement(By.id('textarea')).getAttribute('value');
        if (text == 'Authorization was successful'){
            console.log('Test - SUCCESS');
        }else{
            throw new Error('Test - FAILED');
        }
    } finally {
        await driver.quit();
    }
})()
```

Запустить автотест командой

```
node test.js
```

Результат автотеста в консоли

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a project named 'EXAMPLE-SIMPLE' containing files like 'node_modules', 'package-lock.json', 'package.json', and 'test.js'. The main editor area displays the 'test.js' file content:

```
1 const { Builder, Browser, By, Key, until } = require('selenium-webdriver');
2
3 (async function example() {
4     let driver = await new Builder().forBrowser(Browser.CHROME).build();
5     try {
6         await driver.get('https://somovstudio.github.io/test_eng.html');
7         await driver.findElement(By.name('login')).sendKeys('admin');
8         await driver.findElement(By.name('pass')).sendKeys('0000');
9         await driver.findElement(By.id('buttonLogin')).click();
10        let element = await driver.findElement(By.id('result'));
11        await driver.wait(until.elementIsVisible(element), 5000);
12        let text = await driver.findElement(By.id('textarea')).getAttribute('value');
13        if (text == 'Authorization was successful'){
14            console.log('Test - SUCCESS');
15        }else{
16            throw new Error('Test - FAILED');
17        }
18    } finally {
19        await driver.quit();
20    }
21 })()
```

Below the editor, the terminal tab is active, showing the command line output:

```
PS E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple> node test.js

DevTools listening on ws://127.0.0.1:57980/devtools/browser/5d9e9f81-2b3b-4a99-a083-8dff287ef6dd
Test - SUCCESS
PS E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple>
```

The status bar at the bottom indicates: Ln 2, Col 1 Spaces:4 UTF-8 CRLF () JavaScript

The screenshot shows a Windows Command Prompt window titled 'cmd.exe'. The output shows the same test execution as in VS Code:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple>node test.js

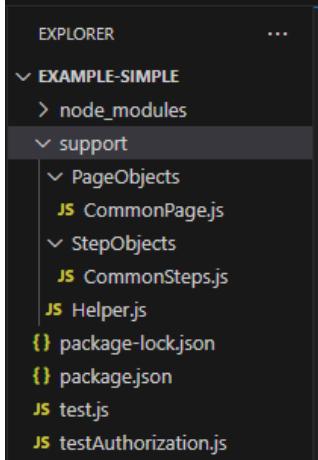
DevTools listening on ws://127.0.0.1:59108/devtools/browser/cea5e3fb-e682-4040-9f90-9f9ecd87b77a
Test - SUCCESS

E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple>
```

Описание паттернов PageObjects и StepObjects

Чтобы построить проект на основе паттернов нужно выполнить следующие действия.

Создать папку support со следующей структурой:



Файл Helper.js - описаны константы

```
module.exports = class Helper {
    static URL = "https://somovstudio.github.io/test_eng.html";
    static LOGIN = "admin";
    static PASSWORD = "0000";
}
```

Файл CommonPage.js - описаны локаторы и статичные методы

```
const { Builder, Browser, By, Key, until } = require('selenium-webdriver');

module.exports = class CommonPage {
    static nameLogin = "login";
    static namePassword = "pass";
    static idButtonLogin = "buttonLogin";
    static idResult = "result";
    static idTextarea = "textarea";

    static async getResultText(driver) {
        let element = await driver.findElement(By.id('result'));
        await driver.wait(until.elementIsVisible(element), 5000);
        let text = await driver.findElement(By.id('textarea')).getAttribute('value');
        return text;
    }
};
```

Файл CommonSteps.js - описан класс методов для выполнения действий

```
const { Builder, Browser, By, Key, until } = require('selenium-webdriver');

module.exports = class CommonSteps {
    constructor(webdriver) {
        this.driver = webdriver;
    }

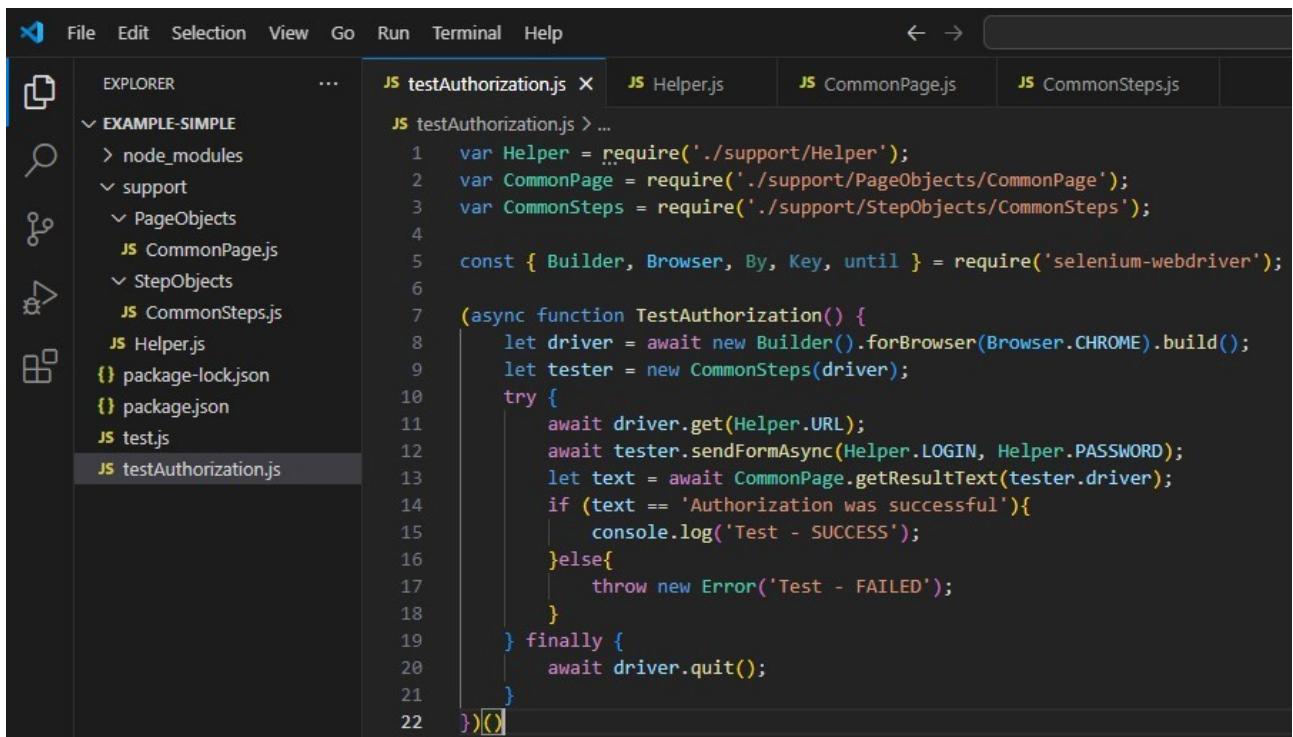
    async sendFormAsync(login, password) {
        await this.driver.findElement(By.name('login')).sendKeys('admin');
        await this.driver.findElement(By.name('pass')).sendKeys('0000');
        await this.driver.findElement(By.id('buttonLogin')).click();
    }
};
```

Файл автотеста testAuthorization.js - используются ранее описанные паттерны

```
var Helper = require('./support/Helper');
var CommonPage = require('./support/PageObjects/CommonPage');
var CommonSteps = require('./support/StepObjects/CommonSteps');

const { Builder, Browser, By, Key, until } = require('selenium-webdriver');

(async function TestAuthorization() {
    let driver = await new Builder().forBrowser(Browser.CHROME).build();
    let tester = new CommonSteps(driver);
    try {
        await driver.get(Helper.URL);
        await tester.sendFormAsync(Helper.LOGIN, Helper.PASSWORD);
        let text = await CommonPage.getResultText(tester.driver);
        if (text == 'Authorization was successful'){
            console.log('Test - SUCCESS');
        }else{
            throw new Error('Test - FAILED');
        }
    } finally {
        await driver.quit();
    }
})()
```



The screenshot shows the Visual Studio Code interface. The left sidebar displays a project structure with a folder named 'EXAMPLE-SIMPLE' containing 'node_modules', 'support' (which includes 'PageObjects' and 'StepObjects' sub-folders), 'Helpers.js', 'package-lock.json', 'package.json', 'test.js', and 'testAuthorization.js'. The main editor area is focused on the 'testAuthorization.js' file, which contains the provided JavaScript code. The code uses the 'selenium-webdriver' library to interact with a browser, specifically Chrome, to perform a login and check for a success message.

Запуск автотест командой:

```
node testAuthorization.js
```

Полезные ссылки:

- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация API Docs
<https://www.selenium.dev/selenium/docs/api/javascript/index.html>
- Официальная страница Visual Studio Code
<https://code.visualstudio.com/>
- Официальная страница NodeJS
<https://nodejs.org/>

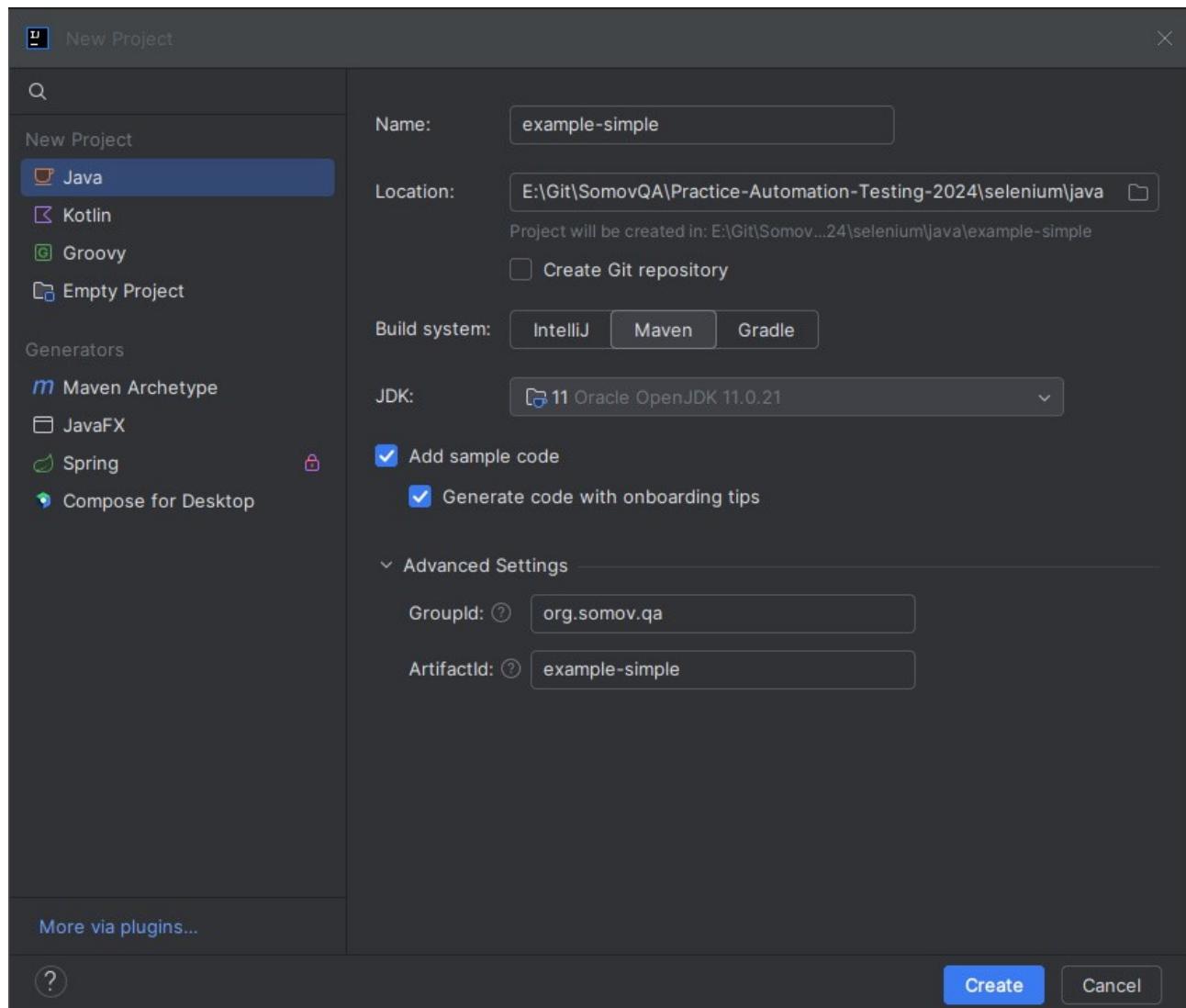
Создание проекта Selenium (Java) с простым автотестом

Для разработки автотестов с помощью Java нужно выполнить следующие действия.

Скачать и установить [IntelliJ IDEA Community Edition](#)

Скачать и установить [Java SE Development Kit 11](#)

В редакторе IntelliJ IDEA создаем новый проект example-simple выбрав Maven и SDK: Oracle OpenJDK 11

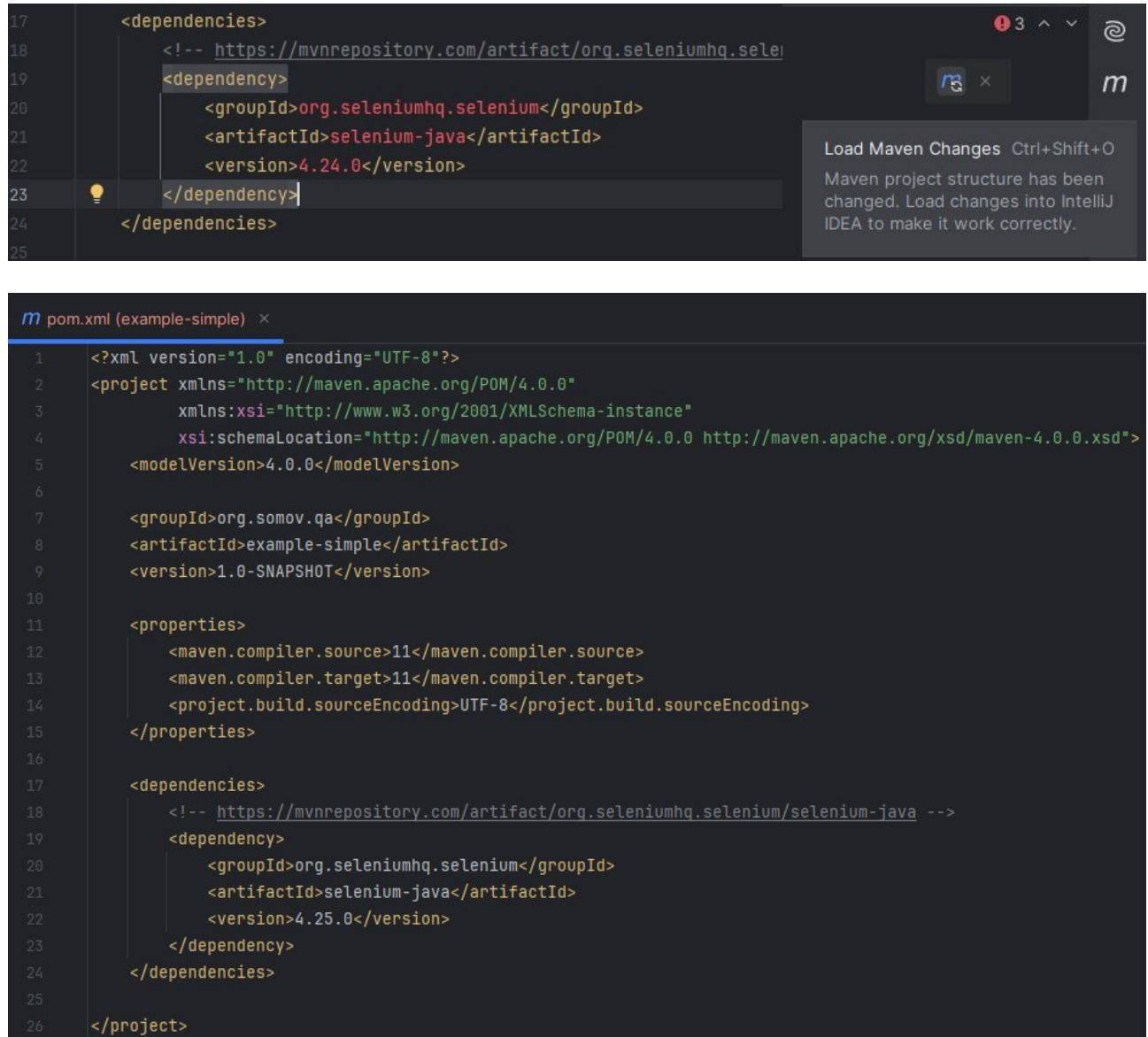


Подключить библиотеку Selenium к проекту в файле pom.xml

ссылка: <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.25.0</version>
    </dependency>
</dependencies>
```

Выполнить загрузку библиотеки в файле pom.xml



```
17     <dependencies>
18         <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/4.24.0 -->
19         <dependency>
20             <groupId>org.seleniumhq.selenium</groupId>
21             <artifactId>selenium-java</artifactId>
22             <version>4.24.0</version>
23         </dependency>
24     </dependencies>
25 
```

Load Maven Changes Ctrl+Shift+O
Maven project structure has been changed. Load changes into IntelliJ IDEA to make it work correctly.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>org.somov.qa</groupId>
8     <artifactId>example-simple</artifactId>
9     <version>1.0-SNAPSHOT</version>
10
11    <properties>
12        <maven.compiler.source>11</maven.compiler.source>
13        <maven.compiler.target>11</maven.compiler.target>
14        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15    </properties>
16
17    <dependencies>
18        <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/4.25.0 -->
19        <dependency>
20            <groupId>org.seleniumhq.selenium</groupId>
21            <artifactId>selenium-java</artifactId>
22            <version>4.25.0</version>
23        </dependency>
24    </dependencies>
25
26 </project>
```

В файле Main.java описать автотест следующим образом

```
WebDriver driver = new ChromeDriver();
driver.get("https://somovstudio.github.io/test_eng.html");
driver.findElement(By.name("login")).sendKeys("admin");
driver.findElement(By.name("pass")).sendKeys("0000");
driver.findElement(By.id("buttonLogin")).click();

WebElement element = driver.findElement(By.id("result"));
Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
wait.until(d -> element.isDisplayed());

String text = driver.findElement(By.id("textarea")).getAttribute("value");
if(text.equals("Authorization was successful")){
    System.out.println("Test - SUCCESS");
} else{
    throw new Error("Test - FAILED");
}

driver.close();
driver.quit();
```

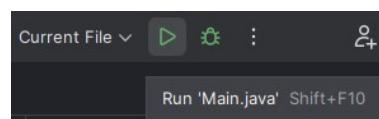
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "example-simple".
- Main.java Content:** The code is as follows:

```
1 package org.somov.qa;
2
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.WebElement;
6 import org.openqa.selenium.chrome.ChromeDriver;
7 import org.openqa.selenium.support.ui.Wait;
8 import org.openqa.selenium.support.ui.WebDriverWait;
9
10 import java.time.Duration;
11 import java.util.Objects;
12
13 public class Main {
14     public static void main(String[] args) {
15         WebDriver driver = new ChromeDriver();
16         driver.get("https://somovstudio.github.io/test_eng.html");
17         driver.findElement(By.name("login")).sendKeys("admin");
18         driver.findElement(By.name("pass")).sendKeys("0000");
19         driver.findElement(By.id("buttonLogin")).click();
20
21         WebElement element = driver.findElement(By.id("result"));
22         Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
23         wait.until(d -> element.isDisplayed());
24
25         String text = driver.findElement(By.id("textarea")).getAttribute("value");
26         if(Objects.equals(text, "Authorization was successful")){
27             System.out.println("Test - SUCCESS");
28         }else{
29             throw new Error("Test - FAILED");
30         }
31
32         driver.close();
33         driver.quit();
34     }
35 }
```

Status Bar: Shows the file path "example-simple > src > main > java > org > somov > qa > Main.java" and the status "1:22 LF UTF-8 4 spaces".

Запустить автотест нажав кнопку Run в IntelliJ IDEA



Результат автотеста в консоли IntelliJ IDEA

The screenshot shows the IntelliJ IDEA "Run" tool window with the following details:

- Run Configuration:** Set to "Main".
- Output Console:** Displays the command "C:\Program Files\Java\jdk-11\bin\java.exe" ... and the message "Test - SUCCESS".
- Status Bar:** Shows the file path "example-simple > src > main > java > org > somov > qa > Main.java" and the status "1:22 LF UTF-8 4 spaces".

Полезные ссылки:

- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация Getting started
https://www.selenium.dev/documentation/webdriver/getting_started/
- Документация Install a Selenium library
https://www.selenium.dev/documentation/webdriver/getting_started/install_library/
- Документация API Docs
<https://www.selenium.dev/selenium/docs/api/java/index.html>
- Официальная страница IntelliJ IDEA Community Edition
<https://www.jetbrains.com/idea/download/other.html>
- Официальная страница Maven
<https://mvnrepository.com/>
- Официальная страница junit5
<https://junit.org/junit5/>
- Официальная страница TestNG
<https://testng.org/>
- Скачать Java SE Development Kit 11
<https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>

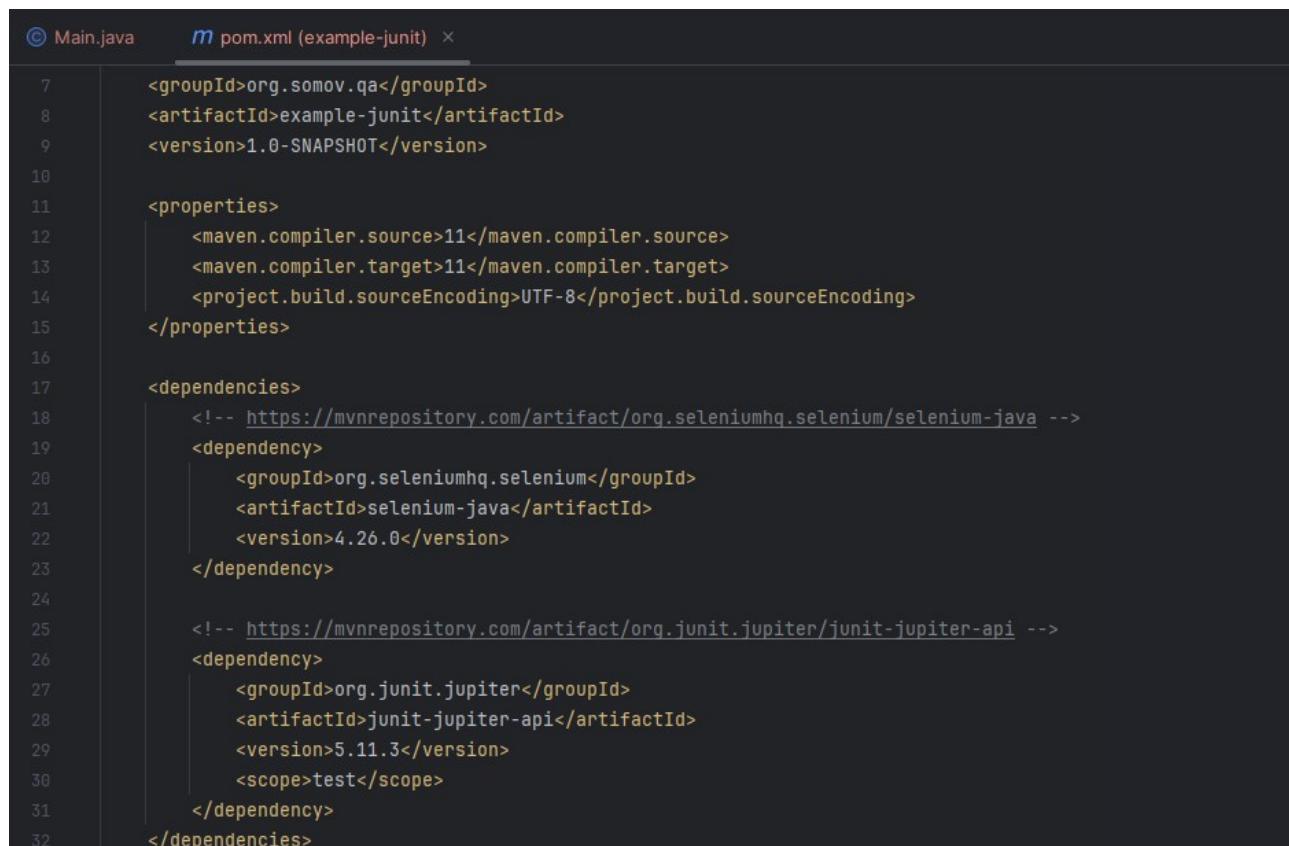
Практика применения JUnit

Чтобы построить проект на основе JUnit нужно выполнить следующие действия.
В редакторе IntelliJ IDEA создаем новый проект example-junit.

Подключить библиотеку Selenium и JUnit Jupiter API к проекту в файле pom.xml
ссылка: <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
ссылка: <https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api>

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.26.0</version>
    </dependency>

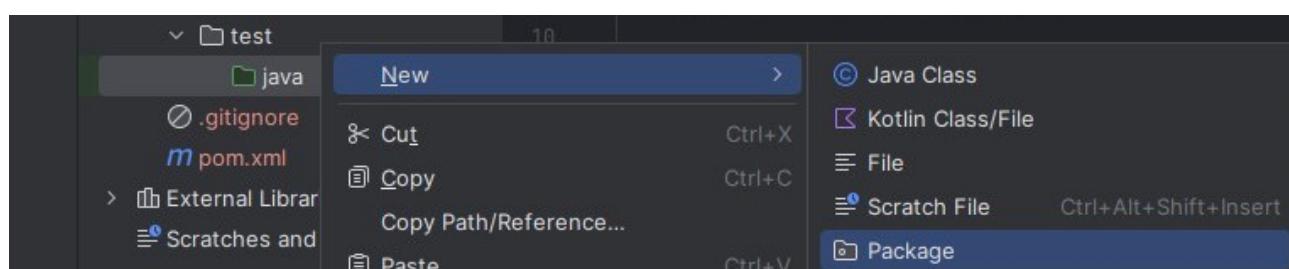
    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.11.3</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```



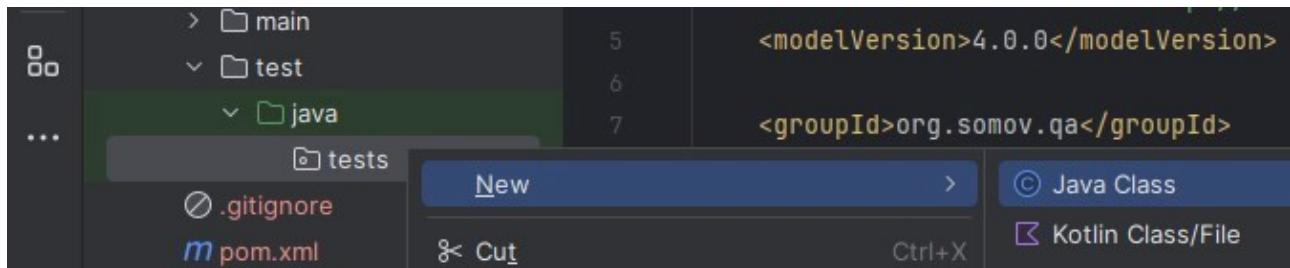
```
>Main.java  pom.xml (example-junit)  ×

7   <groupId>org.somov.qa</groupId>
8   <artifactId>example-junit</artifactId>
9   <version>1.0-SNAPSHOT</version>
10
11  <properties>
12      <maven.compiler.source>11</maven.compiler.source>
13      <maven.compiler.target>11</maven.compiler.target>
14      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15  </properties>
16
17  <dependencies>
18      <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
19      <dependency>
20          <groupId>org.seleniumhq.selenium</groupId>
21          <artifactId>selenium-java</artifactId>
22          <version>4.26.0</version>
23      </dependency>
24
25      <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
26      <dependency>
27          <groupId>org.junit.jupiter</groupId>
28          <artifactId>junit-jupiter-api</artifactId>
29          <version>5.11.3</version>
30          <scope>test</scope>
31      </dependency>
32  </dependencies>
```

В папке test создать пакет tests



В пакете tests создать класс автотеста TestAuthorization.java



Аннотации JUnit 5

JUnit 5 предлагает следующие аннотации для написания тестов.

Аннотации	Описание
@BeforeEach	Аннотированный метод будет запускаться перед каждым тестовым методом в тестовом классе.
@AfterEach	Аннотированный метод будет запускаться после каждого тестового метода в тестовом классе.
@BeforeAll	Аннотированный метод будет запущен перед всеми тестовыми методами в тестовом классе. Этот метод должен быть статическим.
@AfterAll	Аннотированный метод будет запущен после всех тестовых методов в тестовом классе. Этот метод должен быть статическим.
@Test	Он используется, чтобы пометить метод как тест junit.
@DisplayName	Используется для предоставления любого настраиваемого отображаемого имени для тестового класса или тестового метода
@Disable	Он используется для отключения или игнорирования тестового класса или тестового метода из набора тестов.
@Nested	Используется для создания вложенных тестовых классов
@Tag	Пометьте методы тестирования или классы тестов тегами для обнаружения и фильтрации тестов.
@TestFactory	Отметить метод - это тестовая фабрика для динамических тестов.

В файле TestAuthorization.java описать автотест следующим образом

```
package tests;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Tag;
import org.junit.jupiter.api.Test;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.Objects;
```

```

public class TestAuthorization {
    public static WebDriver driver;

    @BeforeAll
    public static void before(){
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

    @Tag("PROD")
    @Test
    public void testAuthorization(){
        driver.get("https://somovstudio.github.io/test_eng.html");
        driver.findElement(By.name("login")).sendKeys("admin");
        driver.findElement(By.name("pass")).sendKeys("0000");
        driver.findElement(By.id("buttonLogin")).click();

        WebElement element = driver.findElement(By.id("result"));
        Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());

        String text = driver.findElement(By.id("textarea")).getAttribute("value");
        System.out.println(text);
        Assertions.assertEquals(text, "Authorization was successful");
    }

    @AfterAll
    public static void after(){
        driver.close();
        driver.quit();
    }
}

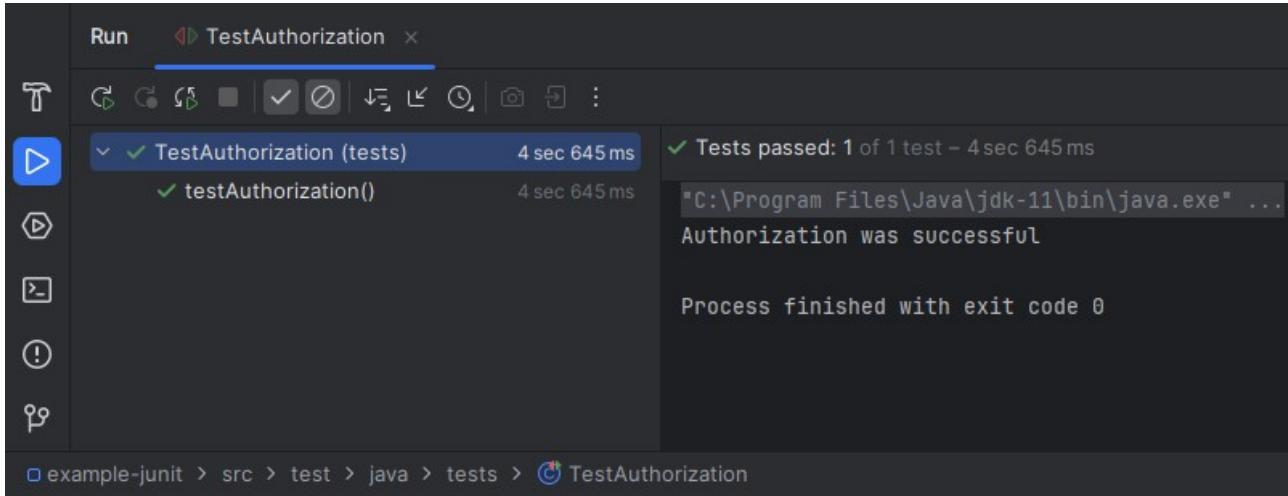
```

```

22  public class TestAuthorization {  Somov Studio *
23      public static WebDriver driver;  10 usages
24
25      @BeforeAll  Somov Studio *
26      public static void before(){
27          driver = new ChromeDriver();
28      }
29
30      @Tag("PROD")  new *
31      @Test
32      public void testAuthorization(){
33          driver.get("https://somovstudio.github.io/test_eng.html");
34          driver.findElement(By.name("login")).sendKeys(...keysToSend: "admin");
35          driver.findElement(By.name("pass")).sendKeys(...keysToSend: "0000");
36          driver.findElement(By.id("buttonLogin")).click();
37
38          WebElement element = driver.findElement(By.id("result"));
39          Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
40          wait.until(d -> element.isDisplayed());
41
42          String text = driver.findElement(By.id("textarea")).getAttribute(name: "value");
43          System.out.println(text);
44          Assertions.assertEquals(text, actual: "Authorization was successful");
45      }
46
47      @AfterAll  Somov Studio *
48      public static void after(){
49          driver.close();
50          driver.quit();
51      }
52  }

```

Запустить автотеста в среде IntelliJ IDEA и убедится что всё работает корректно

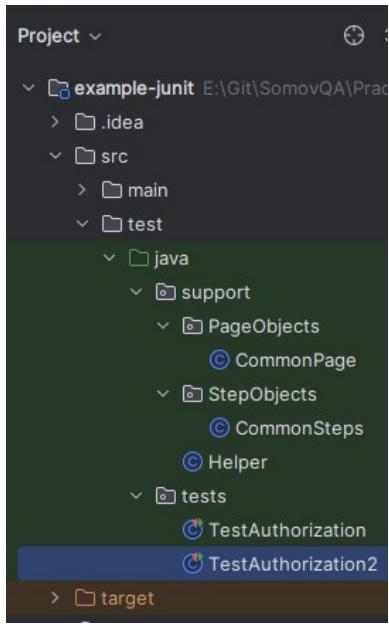


Полезные ссылки:

- Туториал по JUnit 5 - Введение
<https://habr.com/ru/articles/590607/>
- JUnit 5 User Guide
<https://junit.org/junit5/docs/current/user-guide/>
- How to run JUnit5 tests through Command Line
<https://qaautomation.expert/2022/05/12/how-to-run-junit5-tests-through-command-line/>
- Using JUnit 5 Platform
<https://maven.apache.org/surefire/maven-surefire-plugin/examples/junit-platform.html>
- JUnit Jupiter API
<https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api>
- JUnit Jupiter Engine
<https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine>
- Maven Surefire Plugin
<https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin>
- Selenium Java
<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
- Apache Maven
<https://maven.apache.org/download.cgi>

Описание паттернов PageObjects и StepObjects

Описать паттерны PageObjects и StepObjects организовав структуру пакета support следующим образом:



Файл: CommonPage.java - описаны локаторы и статичные методы

```
package support.PageObjects;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;

public class CommonPage {
    public static String nameLogin = "login";
    public static String namePassword = "pass";
    public static String idButtonLogin = "buttonLogin";
    public static String idResult = "result";
    public static String idTextarea = "textarea";

    public static String getResultText(WebDriver driver) {
        WebElement element = driver.findElement(By.id(CommonPage.idResult));
        Wait<WebDriver> wait = new WebDriverWait(driver,
            Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());
        String text = driver.findElement(By.id(CommonPage.idTextarea)).getAttribute("value");
        System.out.println("Get message: " + text);
        return text;
    }
}
```

Файл: CommonSteps.java - описан класс методов для выполнения действий

```
package support.StepObjects;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;
import support.PageObjects.CommonPage;

public class CommonSteps {
    public final WebDriver driver;

    public CommonSteps(WebDriver webdriver) {
        driver = webdriver;
    }

    public void sendForm(String login, String password){
        driver.findElement(
            By.name(CommonPage.nameLogin)).sendKeys(login);
        driver.findElement(
            By.name(CommonPage.namePassword)).sendKeys(password);
        driver.findElement(
            By.id(CommonPage.idButtonLogin)).click();
    }
}
```

Файл Helper.java - описаны константы

```
package support;

public class Helper {
    public static String URL = "https://somovstudio.github.io/test_eng.html";
    public static String LOGIN = "admin";
    public static String PASSWORD = "0000";
}
```

Файл автотеста TestAuthorization2.java - используются ранее описанные паттерны

```
package tests;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Tag;
import org.junit.jupiter.api.Test;

import org.openqa.selenium.chrome.ChromeDriver;

import support.PageObjects.CommonPage;
import support.StepObjects.CommonSteps;
import support.Helper;

public class TestAuthorization2 {

    public static CommonSteps tester;

    @BeforeAll
    public static void before(){
        tester = new CommonSteps(new ChromeDriver());
        tester.driver.manage().window().maximize();
    }

    @Tag("PROD")
    @Test
    public void testAuthorization(){
        tester.driver.get(Helper.URL);
        tester.sendForm(Helper.LOGIN, Helper.PASSWORD);
        String text = CommonPage.getResultText(tester.driver);
        Assertions.assertEquals(text, "Authorization was successful");
    }

    @AfterAll
    public static void after(){
        tester.driver.close();
        tester.driver.quit();
    }
}
```

Запуск автотестов с помощью Maven

Чтобы запустить автотесты с помощью Maven выполнить следующие действия.
Скачать и установить [Apache Maven](#) (архив: apache-maven-3.9.9-bin.zip)

Локальный диск (C:) > Program Files > apache-maven-3.9.9			
Имя	Дата изменения	Тип	Размер
bin	06.11.2024 12:24	Папка с файлами	
boot	06.11.2024 12:24	Папка с файлами	
conf	06.11.2024 12:24	Папка с файлами	
lib	06.11.2024 12:24	Папка с файлами	
LICENSE	14.08.2024 8:48	Файл	19 КБ
NOTICE	14.08.2024 8:48	Файл	5 КБ
README.txt	14.08.2024 8:48	Файл "TXT"	2 КБ

Подключить плагин Maven Surefire Plugin к проекту в файле pom.xml
ссылка: <https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin>
ссылка: <https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine>

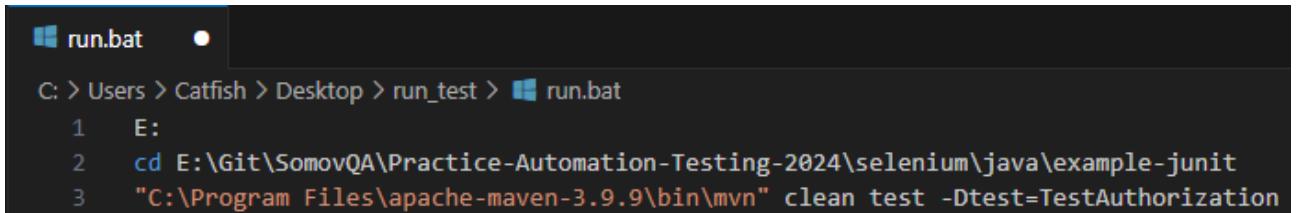
```
<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.26.0</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.11.3</version>
        <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.11.3</version>
        <scope>test</scope>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.5.2</version>
            <dependencies>
                <dependency>
                    <groupId>org.junit.jupiter</groupId>
                    <artifactId>junit-jupiter-engine</artifactId>
                    <version>5.11.3</version>
                </dependency>
            </dependencies>
        </plugin>
    </plugins>
</build>
```

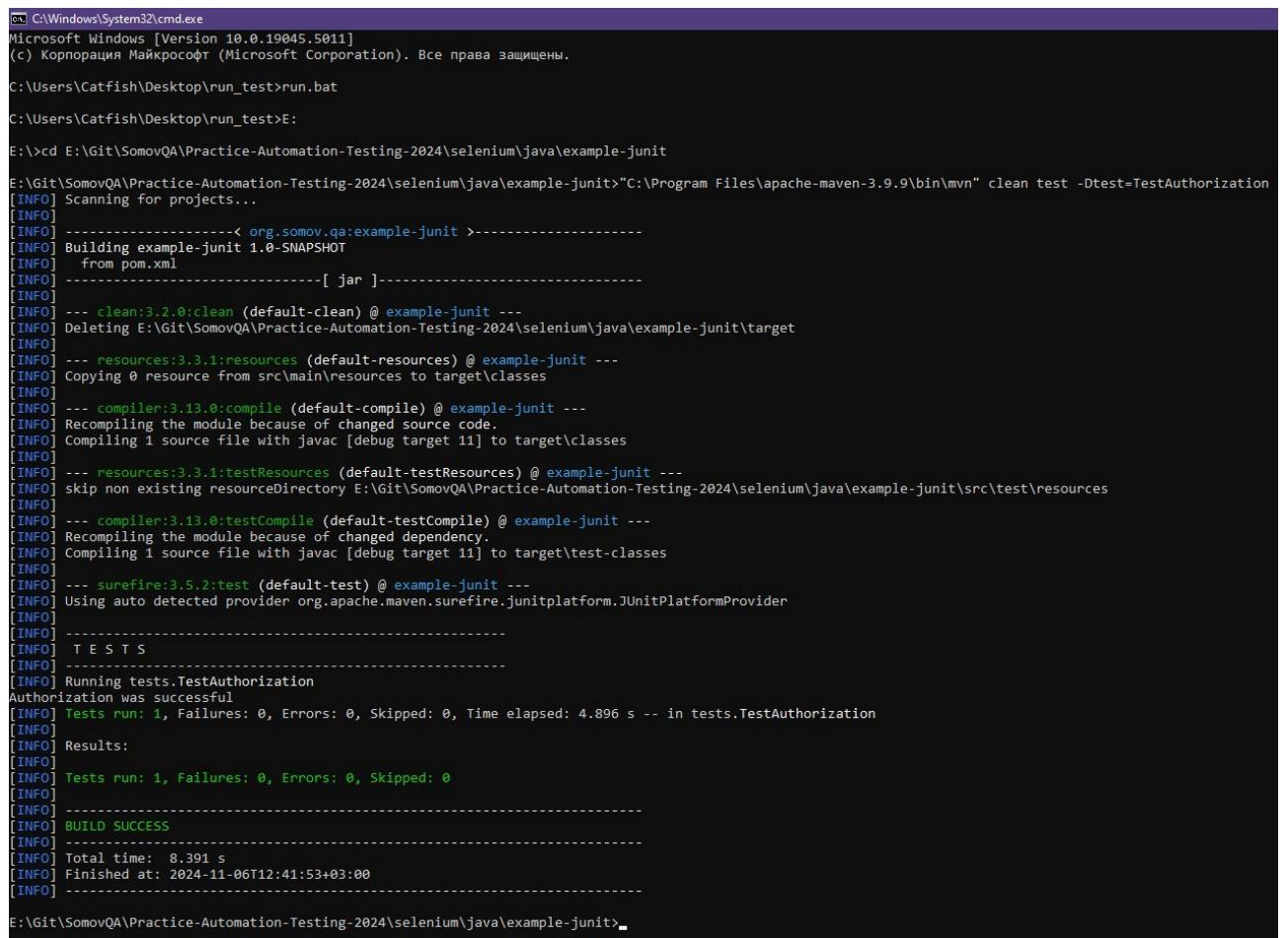
Создать run.bat файл для запуска автотеста из командной строки с помощью Maven

```
E:  
cd E:\Git\...\selenium\java\example-junit  
"C:\Program Files\apache-maven-3.9.9\bin\mvn" clean test -Dtest=TestAuthorization
```



```
C: > Users > Catfish > Desktop > run_test > run.bat  
1 E:  
2 cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit  
3 "C:\Program Files\apache-maven-3.9.9\bin\mvn" clean test -Dtest=TestAuthorization
```

Запустить файл run.bat



```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.19045.5011]  
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.  
C:\Users\Catfish\Desktop\run_test>run.bat  
C:\Users\Catfish\Desktop\run_test>E:  
E:>cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit  
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit>"C:\Program Files\apache-maven-3.9.9\bin\mvn" clean test -Dtest=TestAuthorization  
[INFO] Scanning for projects...  
[INFO] -----< org.somov.qa:example-junit >-----  
[INFO] Building example-junit 1.0-SNAPSHOT  
[INFO]   from pom.xml  
[INFO] -----[ jar ]-----  
[INFO]  
[INFO] --- clean:3.2.0:clean (default-clean) @ example-junit ---  
[INFO] Deleting E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit\target  
[INFO]  
[INFO] --- resources:3.3.1:resources (default-resources) @ example-junit ---  
[INFO] Copying 0 resource from src\main\resources to target\classes  
[INFO]  
[INFO] --- compiler:3.13.0:compile (default-compile) @ example-junit ---  
[INFO] Recompiling the module because of changed source code.  
[INFO] Compiling 1 source file with javac [debug target 11] to target\classes  
[INFO]  
[INFO] --- resources:3.3.1:testResources (default-testResources) @ example-junit ---  
[INFO] skip non existing resourceDirectory E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit\src\test\resources  
[INFO]  
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ example-junit ---  
[INFO] Recompiling the module because of changed dependency.  
[INFO] Compiling 1 source file with javac [debug target 11] to target\test-classes  
[INFO]  
[INFO] --- surefire:3.5.2:test (default-test) @ example-junit ---  
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider  
[INFO]  
[INFO] -----  
[INFO] T E S T S  
[INFO] -----  
[INFO] Running tests.TestAuthorization  
Authorization was successful  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.896 s -- in tests.TestAuthorization  
[INFO]  
[INFO] Results:  
[INFO]  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
[INFO]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 8.391 s  
[INFO] Finished at: 2024-11-06T12:41:53+03:00  
[INFO] -----  
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit>
```

Практика применения TestNG

Чтобы построить проект на основе TestNG нужно выполнить следующие действия.

В редакторе IntelliJ IDEA создаем новый проект example-testng

Подключить библиотеку Selenium и TestNG к проекту в файле pom.xml

ссылка: <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

ссылка: <https://mvnrepository.com/artifact/org.testng/testng>

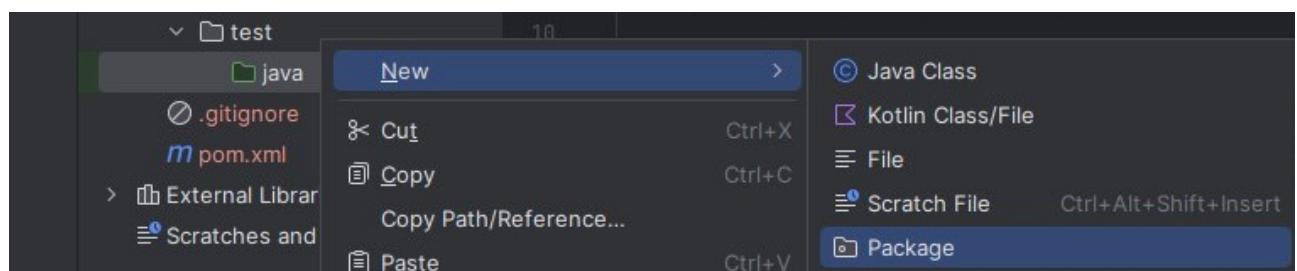
```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.26.0</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.10.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

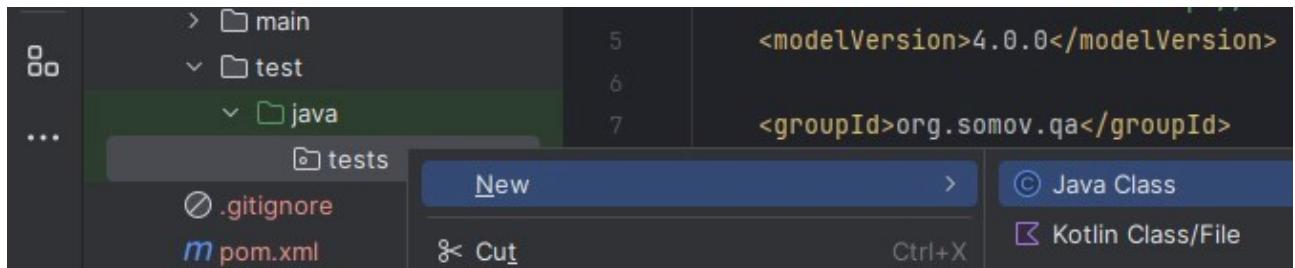
```
17     <dependencies>
18         <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
19         <dependency>
20             <groupId>org.seleniumhq.selenium</groupId>
21             <artifactId>selenium-java</artifactId>
22             <version>4.26.0</version>
23         </dependency>

24
25
26         <!-- https://mvnrepository.com/artifact/org.testng/testng -->
27         <dependency>
28             <groupId>org.testng</groupId>
29             <artifactId>testng</artifactId>
30             <version>7.10.2</version>
31             <scope>test</scope>
32         </dependency>
33     </dependencies>
```

В папке test создать пакет tests



В пакете tests создать класс автотеста TestAuthorization.java



В файле TestAuthorization.java описать автотест следующим образом

```
package tests;

import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.Objects;

public class TestAuthorization {
    public static WebDriver driver;

    @BeforeTest
    public static void before(){
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

    @Test
    public void testAuthorization(){
        driver.get("https://somovstudio.github.io/test_eng.html");
        driver.findElement(By.name("login")).sendKeys("admin");
        driver.findElement(By.name("pass")).sendKeys("0000");
        driver.findElement(By.id("buttonLogin")).click();

        WebElement element = driver.findElement(By.id("result"));
        Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());

        String text = driver.findElement(By.id("textarea")).getAttribute("value");
        System.out.println(text);
        Assert.assertEquals(text, "Authorization was successful");
    }

    @AfterTest
    public static void after(){
        driver.close();
        driver.quit();
    }
}
```

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure under 'example-testng'. The main editor window shows the content of 'TestAuthorization.java'. The code implements a TestNG test for web authorization:

```
public class TestAuthorization {
    public static WebDriver driver; // 11 usages

    @BeforeTest
    public static void before(){
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

    @Test
    public void testAuthorization(){
        driver.get("https://somovstudio.github.io/test_eng.html");
        driver.findElement(By.name("login")).sendKeys(...keysToSend: "admin");
        driver.findElement(By.name("pass")).sendKeys(...keysToSend: "0000");
        driver.findElement(By.id("buttonLogin")).click();

        WebElement element = driver.findElement(By.id("result"));
        Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());

        String text = driver.findElement(By.id("textarea")).getAttribute(name: "value");
        System.out.println(text);
        Assert.assertEquals(text, expected: "Authorization was successful");
    }

    @AfterTest
    public static void after(){
        driver.close();
        driver.quit();
    }
}
```

Запустить автотеста в среде IntelliJ IDEA и убедится что всё работает корректно

The screenshot shows the 'Run' tool window in IntelliJ IDEA. It displays the execution results for the 'TestAuthorization' test:

Test	Time	Status
Default Suite	15 sec 969 ms	✓
example-testng	15 sec 969 ms	✓
TestAuthorization	15 sec 969 ms	✓
testAuthorization	2 sec 864 ms	✓

Details: Tests passed: 1 of 1 test – 15 sec 969 ms
Authorization was successful

Полезные ссылки:

- Selenium Java
<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
- TestNG
<https://mvnrepository.com/artifact/org.testng/testng>

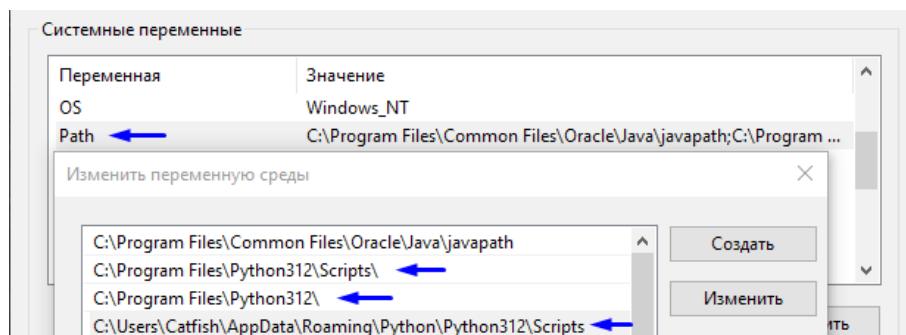
Создание проекта Selenium (Python) с простым автотестом

Для разработки автотестов с помощью Python нужно выполнить следующие действия.

Скачать и установить [Python](#)

В переменной Path прописать путь к Python

(Панель управления > Система > Дополнительные параметры системы > Переменные среды)



```
C:\Program Files\Python312\Scripts\  
C:\Program Files\Python312\  
C:\Users\Имя пользователя\AppData\Roaming\Python\Python312\Scripts\
```

Проверить версию Python, посмотреть установленные библиотеки, обновить pip

```
python -V  
pip list  
python -m pip install --upgrade pip
```

```
Командная строка  
Microsoft Windows [Version 10.0.19045.4894]  
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.  
  
C:\Users\Catfish>python -V  
Python 3.12.6  
  
C:\Users\Catfish>pip list  
Package Version  
----  
pip    24.2  
  
C:\Users\Catfish>python -m pip install --upgrade pip  
Defaulting to user installation because normal site-packages is not writeable  
Requirement already satisfied: pip in c:\program files\python312\lib\site-packages (24.2)  
  
C:\Users\Catfish>
```

Установить Selenium командой
pip install -U selenium

```
Командная строка  
Microsoft Windows [Version 10.0.19045.4894]  
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.  
  
C:\Users\Catfish>pip install -U selenium  
Defaulting to user installation because normal site-packages is not writeable  
Collecting selenium  
  Downloading selenium-4.24.0-py3-none-any.whl.metadata (7.1 kB)  
Collecting urllib3<3,>=1.26 (from selenium)  
  Downloading urllib3-2.2.3-py3-none-any.whl.metadata (6.5 kB)
```

Проверить установленную версию Selenium
pip list

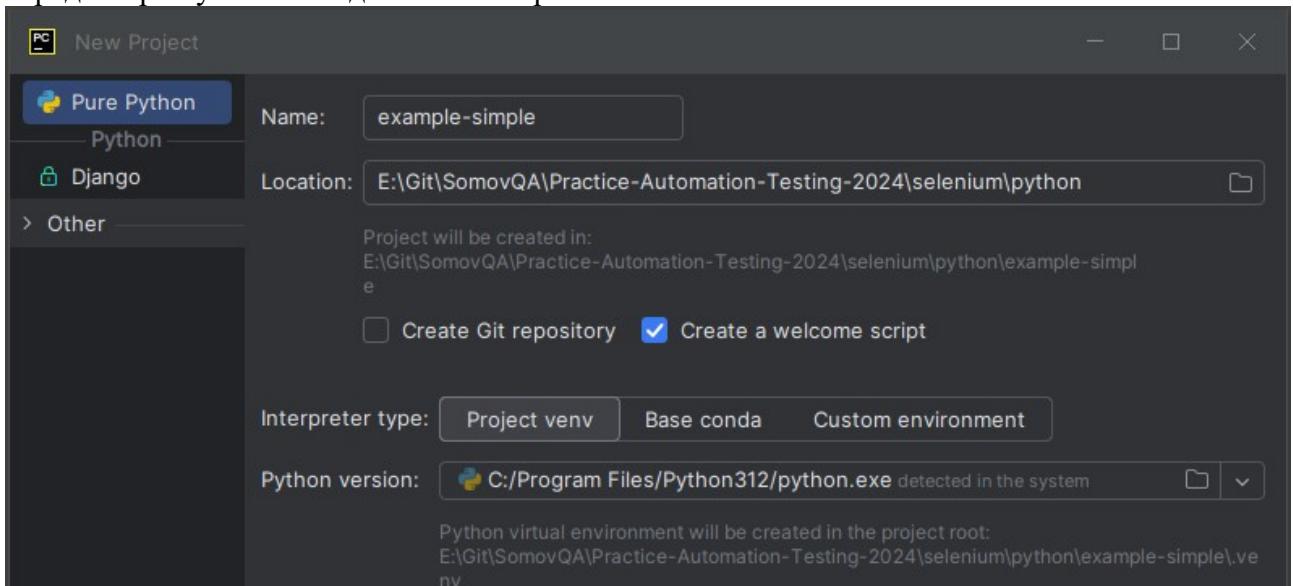
```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>pip list
Package           Version
-----
attrs            24.2.0
certifi          2024.8.30
cffi             1.17.1
h11              0.14.0
idna              3.10
outcome          1.3.0.post0
pip               24.2
pycparser         2.22
PySocks           1.7.1
selenium          4.24.0
sniffio           1.3.1
sortedcontainers  2.4.0
trio              0.26.2
trio-websocket    0.11.1
typing_extensions 4.12.2
urllib3            2.2.3
websocket-client   1.8.0
wsproto            1.2.0

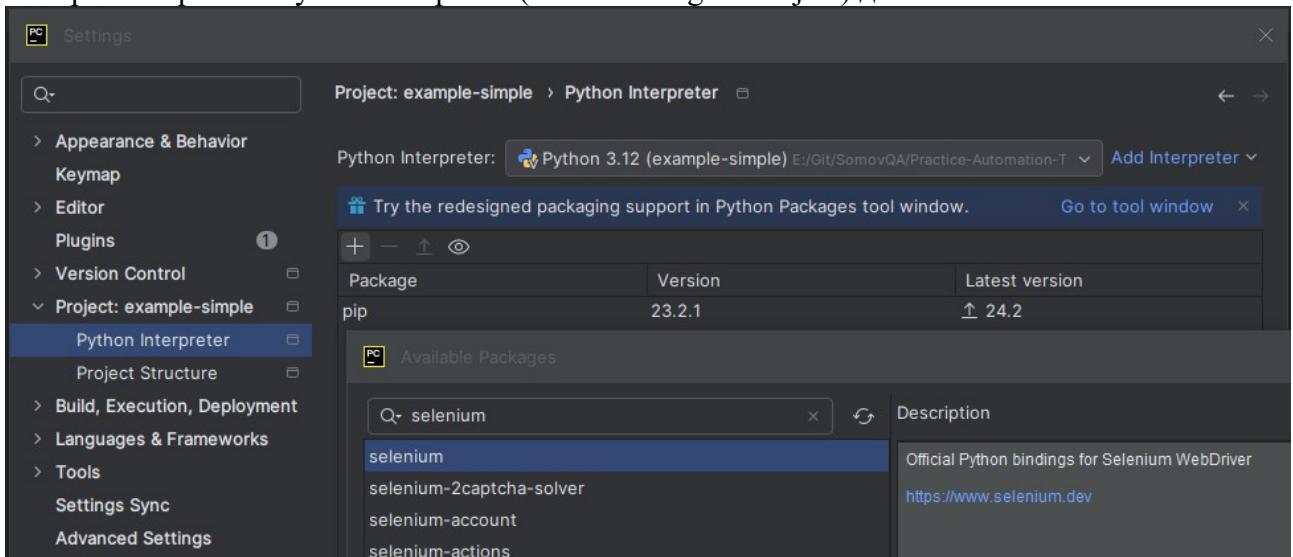
C:\Users\Catfish>
```

Скачать и установить [PyCharm Community Edition](#)

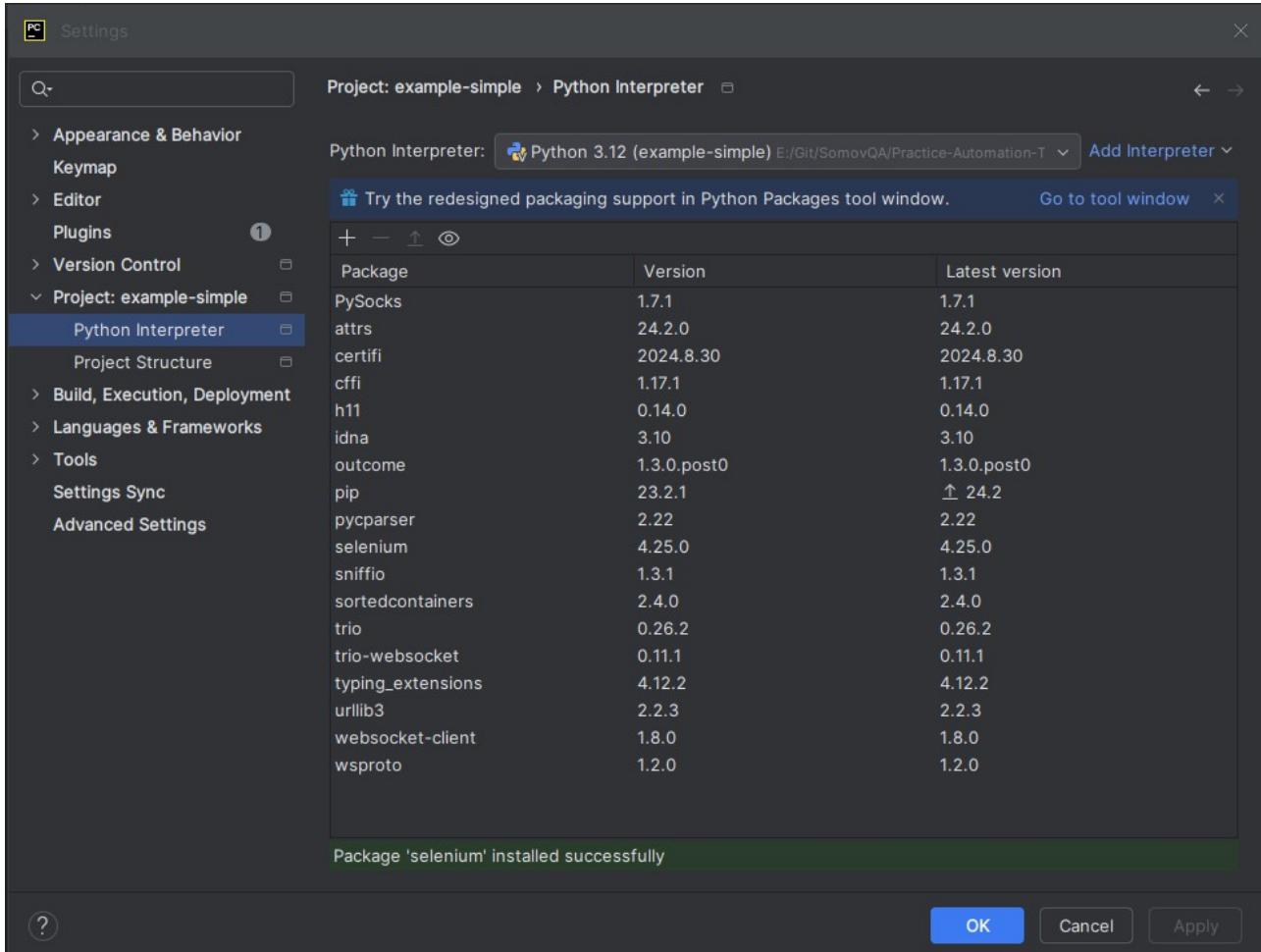
В редакторе PyCharm создаем новый проект



Настройки проекта Python Interpreter (File > Settings > Project) добавить пакет Selenium.



При установке будут добавлены следующие пакеты



В файле main.py описать автотест следующим образом

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

def main():
    driver = webdriver.Chrome()
    driver.get('https://somovstudio.github.io/test_eng.html')
    driver.find_element(By.NAME, 'login').send_keys('admin')
    driver.find_element(By.NAME, 'pass').send_keys('0000')
    driver.find_element(By.ID, 'buttonLogin').click()
    element = driver.find_element(By.ID, 'result')
    wait = WebDriverWait(driver, timeout=5)
    wait.until(lambda d: element.is_displayed())
    text = driver.find_element(By.ID, 'textarea').get_attribute('value')
    if text == 'Authorization was successful':
        print("Test - SUCCESS")
    else:
        print("Test - FAILED")
        raise Exception('Test - FAILED')
    driver.close()
    driver.quit()

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    main()
```

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

def main():  # usage
    driver = webdriver.Chrome()
    driver.get('https://somovstudio.github.io/test_eng.html')
    driver.find_element(By.NAME, value='login').send_keys('admin')
    driver.find_element(By.NAME, value='pass').send_keys('0000')
    driver.find_element(By.ID, value='buttonLogin').click()
    element = driver.find_element(By.ID, value='result')
    wait = WebDriverWait(driver, timeout=5)
    wait.until(lambda d: element.is_displayed())
    text = driver.find_element(By.ID, value='textarea').get_attribute('value')
    if text == 'Authorization was successful':
        print("Test - SUCCESS")
    else:
        print("Test - FAILED")
        raise Exception('Test - FAILED')
    driver.close()
    driver.quit()

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    main()
```

ПРИМЕРЫ: Ожидание отображения элемента на странице

```
time_wait = 25
locator_popup = "//div[@class='popup_body']"
WebDriverWait(driver, time_wait).until(EC.presence_of_element_located((By.XPATH,
    locator_popup)), message=f"Can't find element by locator {locator_popup}")

WebDriverWait(driver, time_wait).until(EC.presence_of_element_located((By.ID,
    "last_order_sended")), message=f"Can't find element by locator last_order_sended")
```

Функции ожидание отображения элемента

```
def find_element(self, locator, time=10):
    return WebDriverWait(self.driver, time).until(EC.presence_of_element_located(locator),
        message=f"Can't find element by locator {locator}")

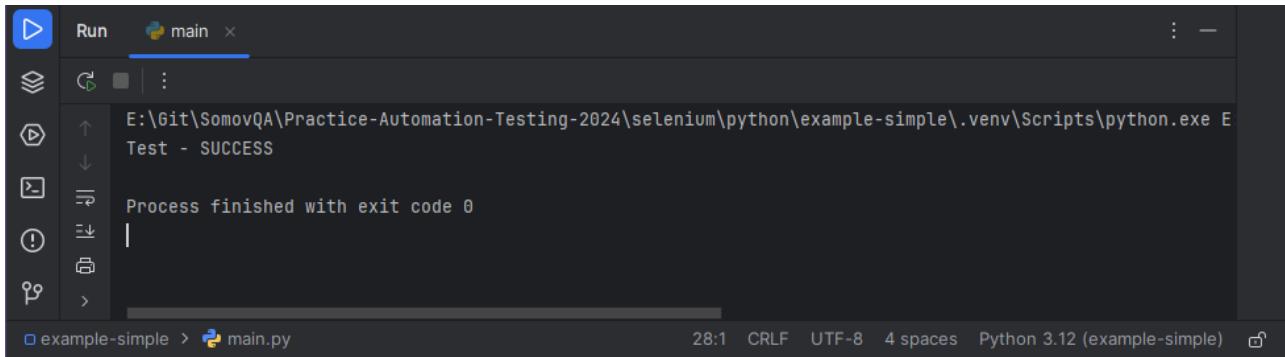
def find_elements(self, locator, time=10):
    return WebDriverWait(self.driver, time).until(EC.presence_of_all_elements_located(locator),
        message=f"Can't find elements by locator {locator}")
```

Простая задержка методом sleep

```
import time

time.sleep(1)
```

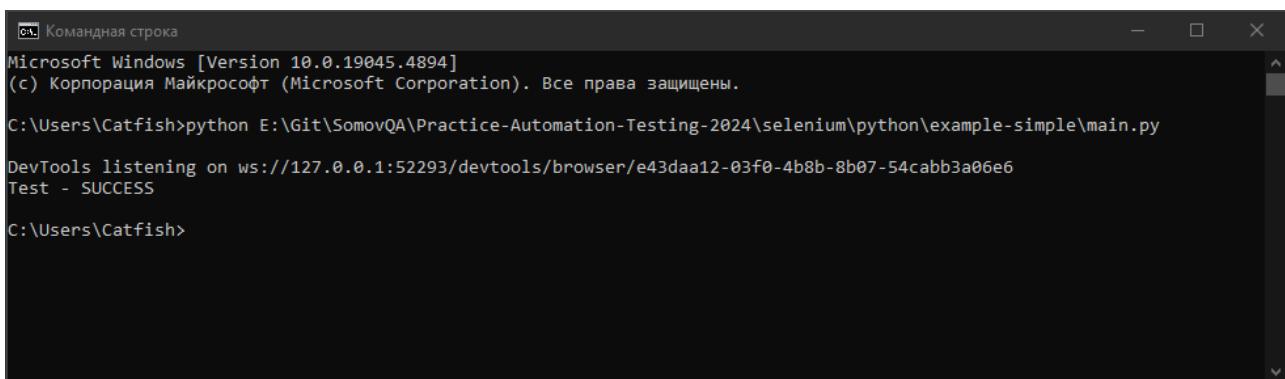
Запуск автотеста в редакторе PyCharm



The screenshot shows the PyCharm interface with the 'Run' tool window open. The current configuration is 'main'. The output pane displays the command run: 'E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-simple\.venv\Scripts\python.exe E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-simple\main.py'. The results show 'Test - SUCCESS' and 'Process finished with exit code 0'. The status bar at the bottom indicates the file is 'main.py', encoding is 'UTF-8', and the Python version is 'Python 3.12 (example-simple)'.

Запуск автотеста из консоли

```
python E:\Git\...\selenium\python\example-simple\main.py
```



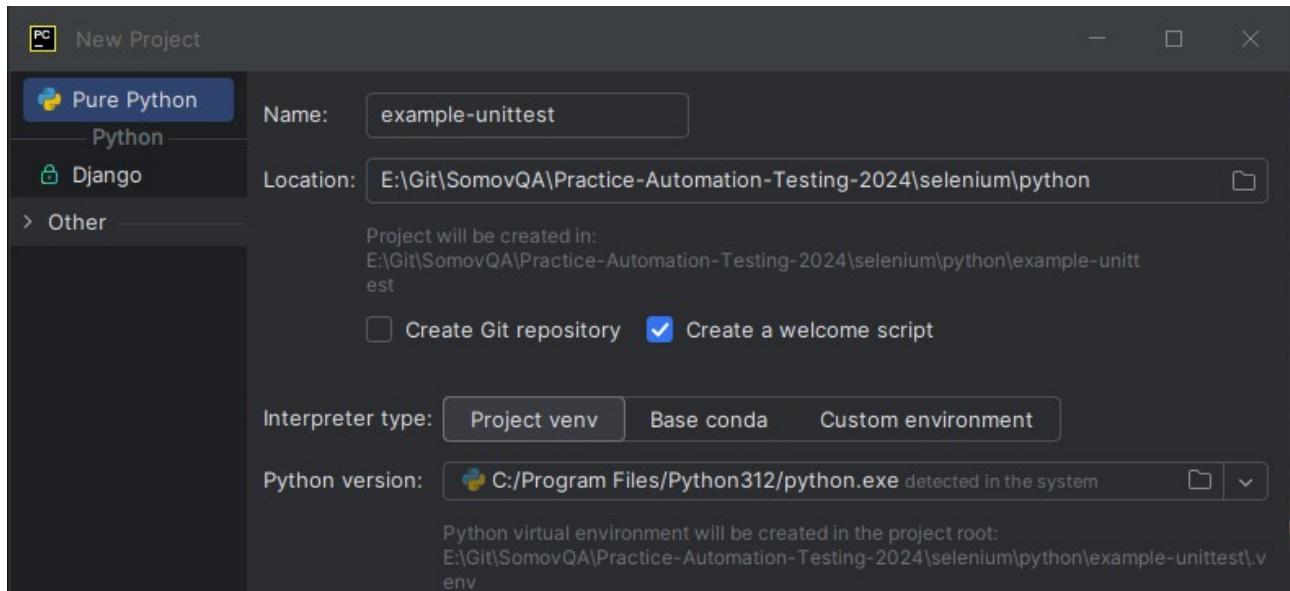
The screenshot shows a Windows Command Prompt window titled 'Командная строка'. It displays the command 'python E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-simple\main.py' being run. The output shows 'DevTools listening on ws://127.0.0.1:52293/devtools/browser/e43daa12-03f0-4b8b-8b07-54cabb3a06e6' and 'Test - SUCCESS'. The prompt then changes to 'C:\Users\Catfish>'.

Полезные ссылки:

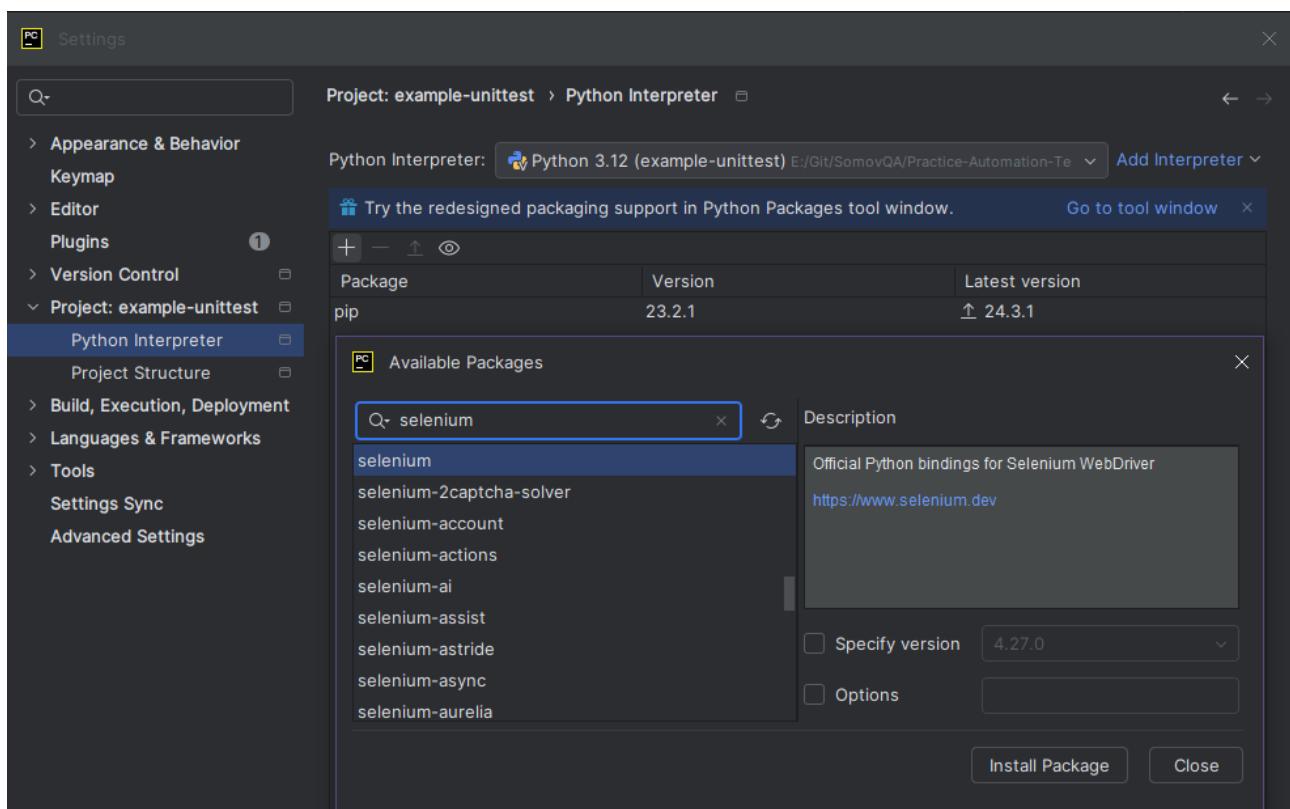
- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация API Docs (python)
<https://www.selenium.dev/selenium/docs/api/py/index.html>
- Официальная страница PyCharm Community Edition
<https://www.jetbrains.com/pycharm/download/other.html>
- Официальная страница Python
<https://www.python.org/>

Практика применения Unittest (Python)

Чтобы построить проект на основе Unittest нужно выполнить следующие действия.
В редакторе PyCharm создаем новый проект

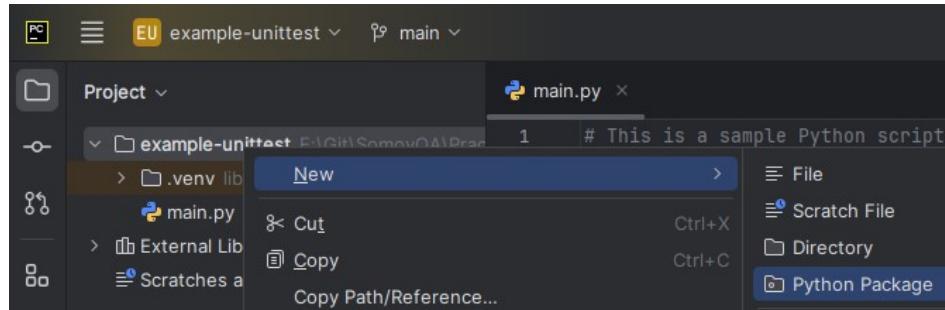


Настройки проекта Python Interpreter (File > Settings > Project) добавить пакет Selenium.

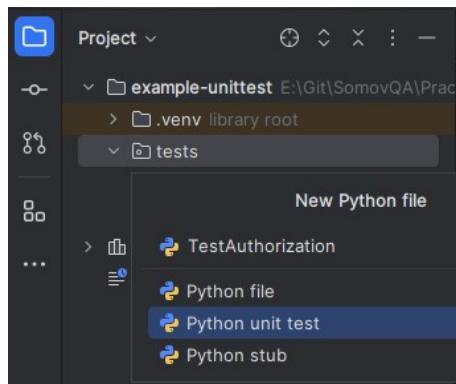


При установке будут добавлены пакеты Selenium

Создать пакет с именем tests



В пакете tests создать модульный тест (Python unit test) с именем TestAuthorization.py



В файле TestAuthorization.py описать автотест следующим образом

```
import unittest

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

class MyTestCase(unittest.TestCase):
    def test_something(self):
        driver = webdriver.Chrome()
        driver.maximize_window()
        driver.get('https://somovstudio.github.io/test_eng.html')
        driver.find_element(By.NAME, 'login').send_keys('admin')
        driver.find_element(By.NAME, 'pass').send_keys('0000')
        driver.find_element(By.ID, 'buttonLogin').click()
        element = driver.find_element(By.ID, 'result')
        wait = WebDriverWait(driver, timeout=5)
        wait.until(lambda d: element.is_displayed())
        text = driver.find_element(By.ID, 'textarea').get_property('value')
        print("Get message: " + text)

        self.assertEqual(text, 'Authorization was successful')

        driver.close()
        driver.quit()

if __name__ == '__main__':
    unittest.main()
```

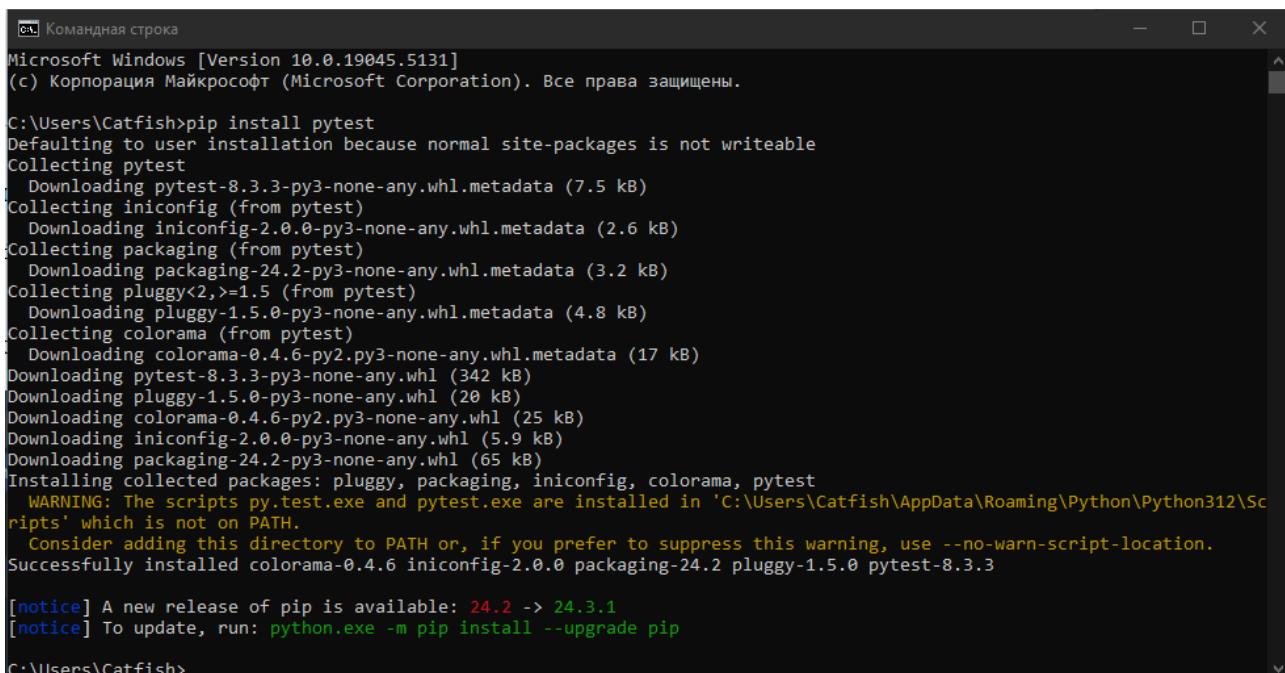
Запуск автотеста в редакторе PyCharm или в консоли командой:

```
python E:\Git\...\selenium\python\example-unittest\tests\TestAuthorization.py
```

Практика применения PyTest (Python)

Чтобы построить проект на основе PyTest нужно выполнить следующие действия.
Установить PyTest

```
pip install pytest  
pip3 install pytest
```

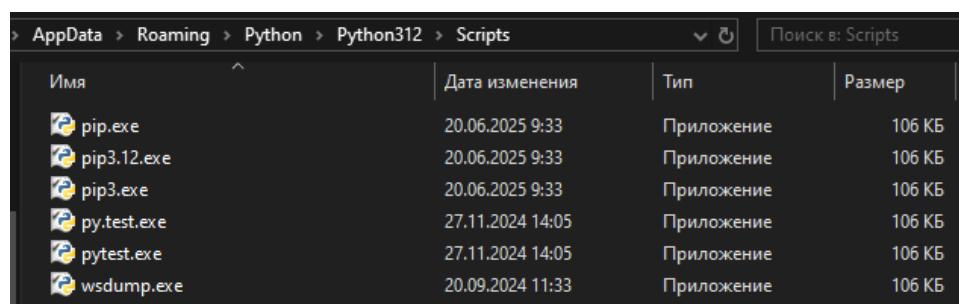


```
Командная строка  
Microsoft Windows [Version 10.0.19045.5131]  
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.  
  
C:\Users\Catfish>pip install pytest  
Defaulting to user installation because normal site-packages is not writeable  
Collecting pytest  
  Downloading pytest-8.3.3-py3-none-any.whl.metadata (7.5 kB)  
Collecting iniconfig (from pytest)  
  Downloading iniconfig-2.0.0-py3-none-any.whl.metadata (2.6 kB)  
Collecting packaging (from pytest)  
  Downloading packaging-24.2-py3-none-any.whl.metadata (3.2 kB)  
Collecting pluggy<2,>=1.5 (from pytest)  
  Downloading pluggy-1.5.0-py3-none-any.whl.metadata (4.8 kB)  
Collecting colorama (from pytest)  
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)  
Downloading pytest-8.3.3-py3-none-any.whl (342 kB)  
Downloading pluggy-1.5.0-py3-none-any.whl (20 kB)  
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)  
Downloading iniconfig-2.0.0-py3-none-any.whl (5.9 kB)  
Downloading packaging-24.2-py3-none-any.whl (65 kB)  
Installing collected packages: pluggy, packaging, iniconfig, colorama, pytest  
  WARNING: The scripts py.test.exe and pytest.exe are installed in 'C:\Users\Catfish\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.  
    Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.  
Successfully installed colorama-0.4.6 iniconfig-2.0.0 packaging-24.2 pluggy-1.5.0 pytest-8.3.3  
  
[notice] A new release of pip is available: 24.2 -> 24.3.1  
[notice] To update, run: python.exe -m pip install --upgrade pip  
  
C:\Users\Catfish>
```

Выполнить обновление если это было предложено

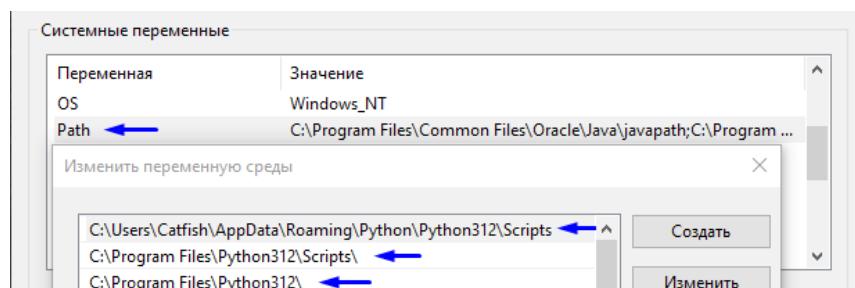
```
python.exe -m pip install --upgrade pip
```

PyTest будет установлен по адресу: C:\Users\Имя\AppData\Roaming\Python\Python312\Scripts



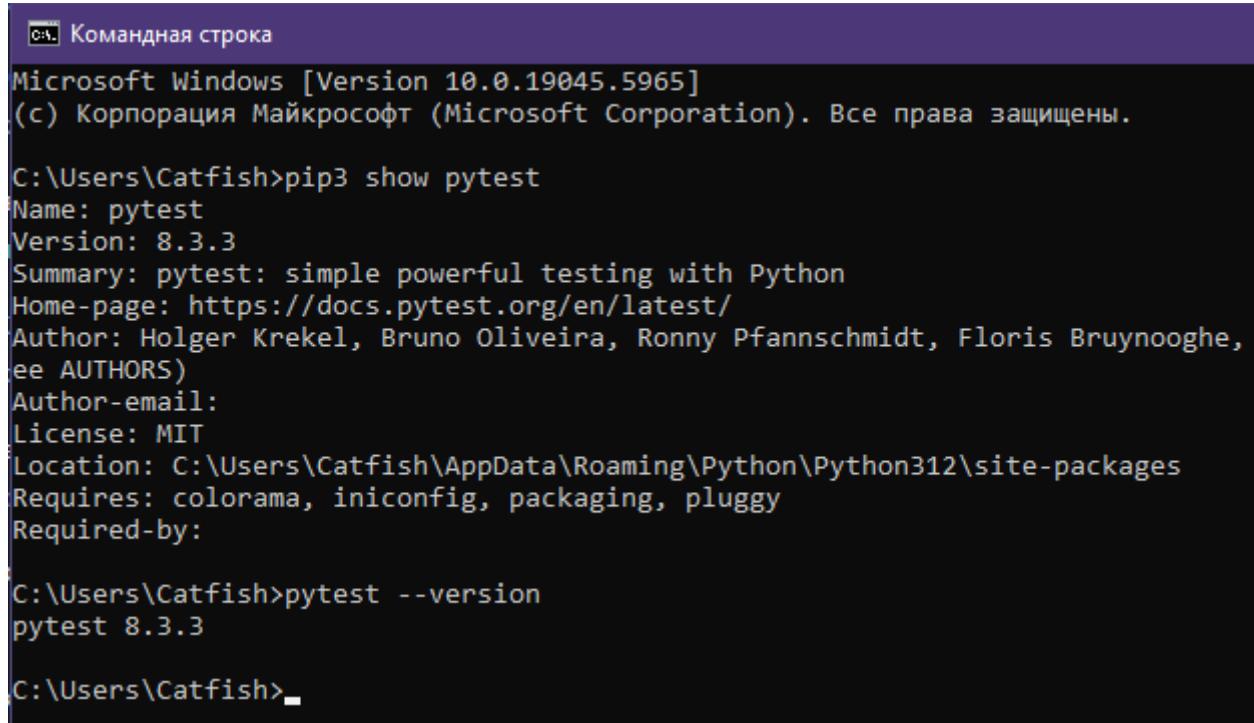
Имя	Дата изменения	Тип	Размер
pip.exe	20.06.2025 9:33	Приложение	106 КБ
pip3.12.exe	20.06.2025 9:33	Приложение	106 КБ
pip3.exe	20.06.2025 9:33	Приложение	106 КБ
py.test.exe	27.11.2024 14:05	Приложение	106 КБ
pytest.exe	27.11.2024 14:05	Приложение	106 КБ
wsdump.exe	20.09.2024 11:33	Приложение	106 КБ

Добавить адрес C:\Users\Имя\AppData\Roaming\Python\Python312\Scripts в системную переменную среды Path



Для проверки выполнить команду

```
pip3 show pytest
pytest --version
```



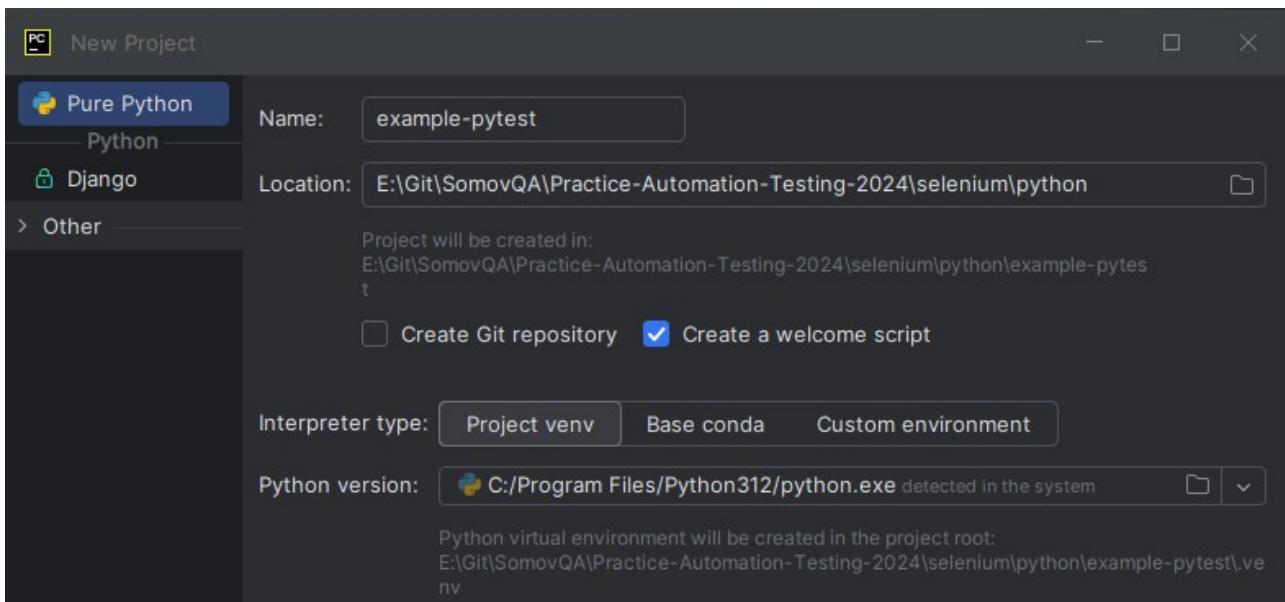
```
Командная строка
Microsoft Windows [Version 10.0.19045.5965]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>pip3 show pytest
Name: pytest
Version: 8.3.3
Summary: pytest: simple powerful testing with Python
Home-page: https://docs.pytest.org/en/latest/
Author: Holger Krekel, Bruno Oliveira, Ronny Pfannschmidt, Floris Bruynooghe, et al. (see AUTHORS)
Author-email:
License: MIT
Location: C:\Users\Catfish\AppData\Roaming\Python\Python312\site-packages
Requires: colorama, configparser, packaging, pluggy
Required-by:

C:\Users\Catfish>pytest --version
pytest 8.3.3

C:\Users\Catfish>
```

В редакторе PyCharm создаем новый проект

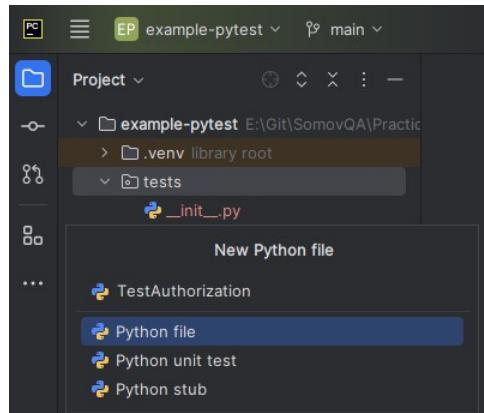
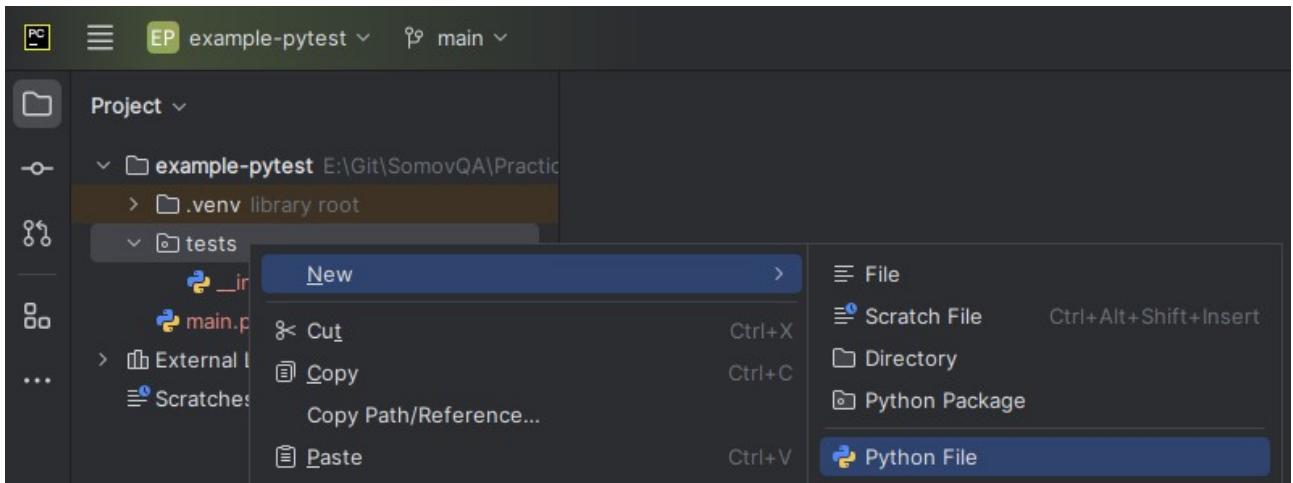


Настройки проекта Python Interpreter (File > Settings > Project) добавить пакет Selenium.

При установке будут добавлены пакеты Selenium

В корне проекта создать пакет с именем tests

В пакете tests создать файл (Python file) с именем TestAuthorization.py

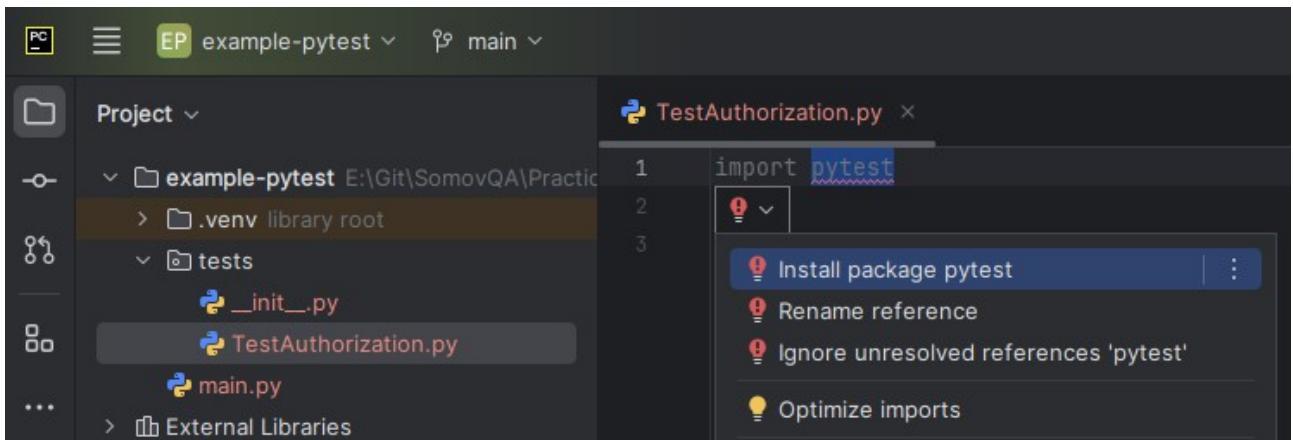


В файле TestAuthorization.py ввести строку

```
import pytest
```

Выполнить установку пакета pytest как предполагает редактор или через настройки проекта

Python Interpreter (File > Settings > Project)



После установки pytest.exe будет находиться по адресу \example-pytest\.venv\Scripts

Practice-Automation-Testing-2024 > selenium > python > example-pytest > .venv > Scripts >			
Имя	Дата изменения	Тип	Размер
.pytest_cache	27.11.2024 14:51	Папка с файлами	
activate	27.11.2024 12:48	Файл	3 КБ
activate.bat	27.11.2024 12:48	Пакетный файл ...	2 КБ
activate.fish	27.11.2024 12:48	Файл "FISH"	4 КБ
activate.nu	27.11.2024 12:48	Файл "NU"	3 КБ
activate.ps1	27.11.2024 12:48	Сценарий Windo...	2 КБ
activate_this.py	27.11.2024 12:48	JetBrains PyChar...	2 КБ
deactivate.bat	27.11.2024 12:48	Пакетный файл ...	1 КБ
pip.exe	27.11.2024 12:48	Приложение	106 КБ
pip3.12.exe	27.11.2024 12:48	Приложение	106 КБ
pip-3.12.exe	27.11.2024 12:48	Приложение	106 КБ
pip3.exe	27.11.2024 12:48	Приложение	106 КБ
py.test.exe	27.11.2024 13:20	Приложение	106 КБ
pydoc.bat	27.11.2024 12:48	Пакетный файл ...	1 КБ
pytest.exe	27.11.2024 13:20	Приложение	106 КБ
python.exe	27.11.2024 12:48	Приложение	264 КБ
pythonw.exe	27.11.2024 12:48	Приложение	253 КБ
wsdump.exe	27.11.2024 12:55	Приложение	106 КБ

В файле TestAuthorization.py описать автотест следующим образом

```
import pytest

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

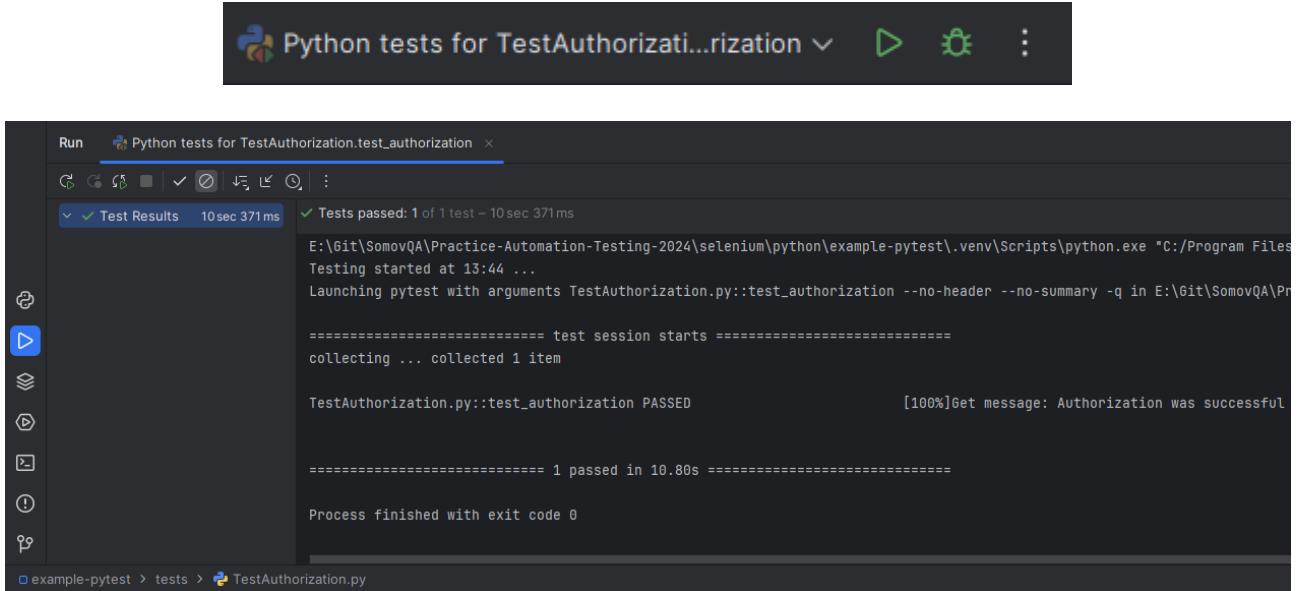
def test_authorization():
    driver = webdriver.Chrome()
    driver.maximize_window()
    driver.get('https://somovstudio.github.io/test_eng.html')
    driver.find_element(By.NAME, 'login').send_keys('admin')
    driver.find_element(By.NAME, 'pass').send_keys('0000')
    driver.find_element(By.ID, 'buttonLogin').click()
    element = driver.find_element(By.ID, 'result')
    wait = WebDriverWait(driver, timeout=5)
    wait.until(lambda d: element.is_displayed())
    text = driver.find_element(By.ID, 'textarea').get_property('value')
    print("Get message: " + text)

    assert text == 'Authorization was successful', "Получено некорректное сообщение"

    driver.close()
    driver.quit()

if __name__ == '__main__':
    pytest.main(["-s", "TestAuthorization.py"])
```

Запуск автотеста в редакторе PyCharm



```
Python tests for TestAuthorization.test_authorization x
Run Python tests for TestAuthorization.test_authorization x
Test Results 10 sec 371ms
Tests passed: 1 of 1 test - 10 sec 371ms
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts\python.exe "C:/Program Files/Testing started at 13:44 ...
Launching pytest with arguments TestAuthorization.py::test_authorization --no-header --no-summary -q in E:\Git\SomovQA\Pr
=====
collecting ... collected 1 item

TestAuthorization.py::test_authorization PASSED [100%]Get message: Authorization was successful

=====
1 passed in 10.80s =====
Process finished with exit code 0
```

ИЛИ В КОНСОЛИ ВЫПОЛНИТЬ КОМАНДЫ

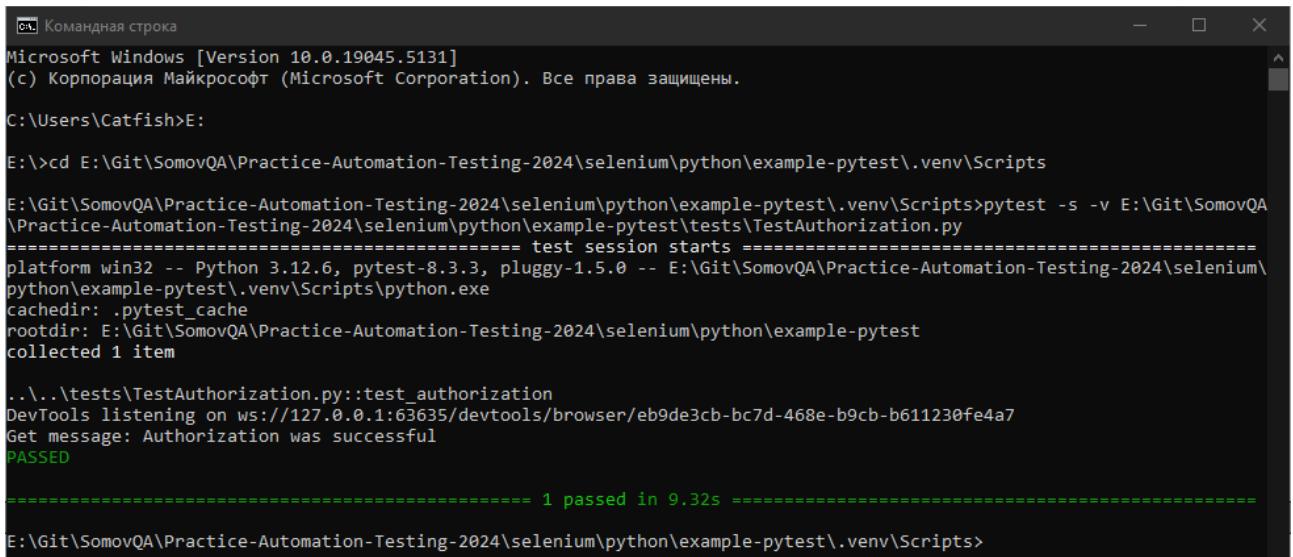
```
pytest -s -v E:\Git\...\selenium\python\example-pytest\tests\TestAuthorization.py
```

МОЖНО ИСПОЛЬЗОВАТЬ PyTest из папки .venv/Scripts

E:

```
cd E:\Git\...\selenium\python\example-pytest\.venv\Scripts
```

```
pytest -s -v E:\Git\...\selenium\python\example-pytest\tests\TestAuthorization.py
```



```
Командная строка
Microsoft Windows [Version 10.0.19045.5131]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

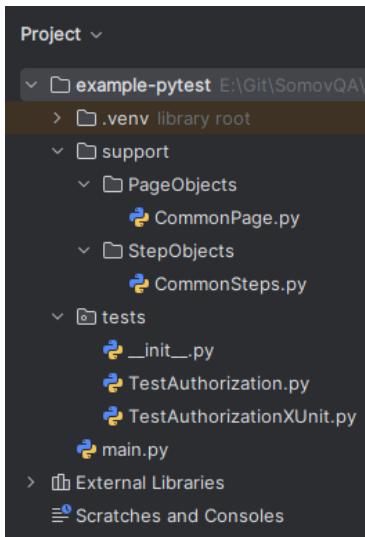
C:\Users\Catfish>E:
E:>cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts>pytest -s -v E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\tests\TestAuthorization.py
=====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest
collected 1 item

..\tests\TestAuthorization.py::test_authorization
DevTools listening on ws://127.0.0.1:63635/devtools/browser/eb9de3cb-bc7d-468e-b9cb-b611230fe4a7
Get message: Authorization was successful
PASSED

=====
1 passed in 9.32s =====
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts>
```

Практика PyTest в стиле xUnit

Описание паттернов PageObjects и StepObjects



Файл CommonPage.py - описаны локаторы и статичные методы

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

class CommonPage:
    nameLogin = "login"
    namePassword = "pass"
    idButtonLogin = "buttonLogin"
    idResult = "result"
    idTextarea = "textarea"

    def getResultText(driver):
        element = driver.find_element(By.ID, CommonPage.idResult)
        wait = WebDriverWait(driver, timeout=5)
        wait.until(lambda d: element.is_displayed())
        text = driver.find_element(By.ID,
                                   CommonPage.idTextarea).get_property('value')
        print("Get message: " + text)
        return text
```

Файл CommonSteps.py - описан класс методов для выполнения действий

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

from support.PageObjects.CommonPage import CommonPage

class CommonSteps:

    def __init__(self, webdriver):
        self.driver = webdriver

    def sendForm(self, login, password):
        self.driver.find_element(By.NAME,
                               CommonPage.nameLogin).send_keys(login)
        self.driver.find_element(By.NAME,
                               CommonPage.namePassword).send_keys(password)
        self.driver.find_element(By.ID,
                               CommonPage.idButtonLogin).click()
```

Файл автотеста TestAuthorizationXUnit.py - используются ранее описанные паттерны

```
import pytest

from selenium import webdriver
from support.PageObjects.CommonPage import CommonPage
from support.StepObjects.CommonSteps import CommonSteps

class TestAuthorizationXUnit:

    def setup_method(self):
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()

    def test(self):
        tester = CommonSteps(self.driver)
        tester.driver.get('https://somovstudio.github.io/test_eng.html')
        tester.sendForm( 'admin', '0000')
        text = CommonPage.getResultText(tester.driver)
        assert text == 'Authorization was successful', "Получено некорректное сообщение"
```

```

def teardown_method(self):
    self.driver.close()
    self.driver.quit()

if __name__ == '__main__':
    pytest.main(["-s", "TestAuthorizationXUnit.py"])

```

Запуск автотеста в редакторе PyCharm

The screenshot shows the PyCharm interface with the 'Run' tab selected. A single test run is listed: 'Python tests in TestAuthorizationXUnit.py'. The results show 'Tests passed: 1 of 1 test - 8 sec 174 ms'. The terminal output pane displays the command 'E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts\python.exe' and the test session log. The log indicates a successful test named 'test' from 'TestAuthorizationXUnit.py' that passed in 13.23s. The message 'Get message: Authorization was successful' is shown. The process finished with exit code 0.

ИЛИ В КОНСОЛИ ВЫПОЛНИТЬ КОМАНДЫ

Е:

```

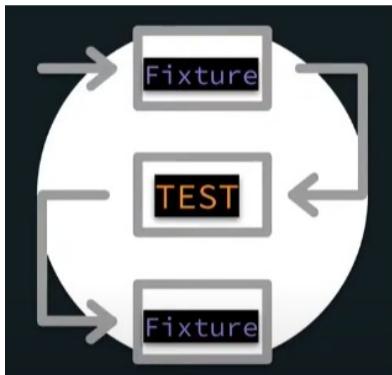
cd E:\Git\...\selenium\python\example-pytest\.venv\Scripts
pytest -s -v E:\Git\...\selenium\python\example-
pytest\tests\TestAuthorizationXUnit.py

```

The screenshot shows a terminal window titled 'Командная строка' (Command Line). The user navigates to the directory 'E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts'. The command 'pytest -s -v E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\tests\TestAuthorizationXUnit.py' is run. The terminal output shows the test session starting on 'platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts\python.exe'. It then lists the collected test item '...\\tests\\TestAuthorizationXUnit.py::test'. The test passes with the message 'Get message: Authorization was successful'. The final status is 'PASSED'. The message '1 passed in 9.53s' is displayed at the end.

Практика PyTest с применением Fixture

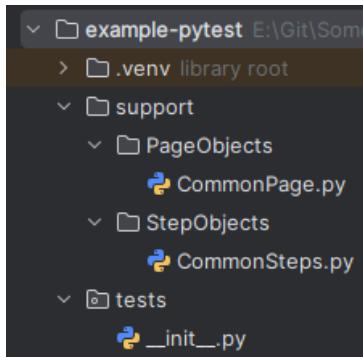
Фикстуры помогают сократить дублирующийся код



Порядок и частота вызова фикстур

1. session - один раз при запуске всех тестов
 2. module - один раз в пакете
 3. class - перед тестовым классом
 4. function - перед каждым тестом

Описание паттернов PageObjects и StepObjects



В файле CommonPage.py

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

class CommonPage:
    nameLogin = "login"
    namePassword = "pass"
    idButtonLogin = "buttonLogin"
    idResult = "result"
    idTextarea = "textarea"

    def getResultText(driver):
        element = driver.find_element(By.ID, CommonPage.idResult)
        wait = WebDriverWait(driver, timeout=5)
        wait.until(lambda d: element.is_displayed())
        text = driver.find_element(By.ID,
                                   CommonPage.idTextarea).get_property('value')
        print("Get message: " + text)
        return text
```

В файле CommonSteps.py

В файле TestAuthorizationFixture1.py описать автотест с использованием фикстур

```
import pytest

from selenium import webdriver
from support.PageObjects.CommonPage import CommonPage
from support.StepObjects.CommonSteps import CommonSteps

@pytest.fixture(scope="function")
def init_driver():
    driver = webdriver.Chrome()
    driver.maximize_window()
    return driver

def test(init_driver):
    tester = CommonSteps(init_driver)
    tester.driver.get('https://somovstudio.github.io/test_eng.html')
    tester.sendForm( 'admin', '0000')
    text = CommonPage.getResultText(tester.driver)
    assert text == 'Authorization was successful', "Получено некорректное сообщение"
    tester.driver.close()
    tester.driver.quit()
```

В файле TestAuthorizationFixture2.py использование фикстур как Setup, Test, Teardown

строка yield driver - останавливает выполнение функции init_driver, запускает тест test_authorization передает в него driver и после завершения тест возвращает управление в функцию init_driver где происходит закрытие браузера независимо от того успешно ли был пройден тест или нет.

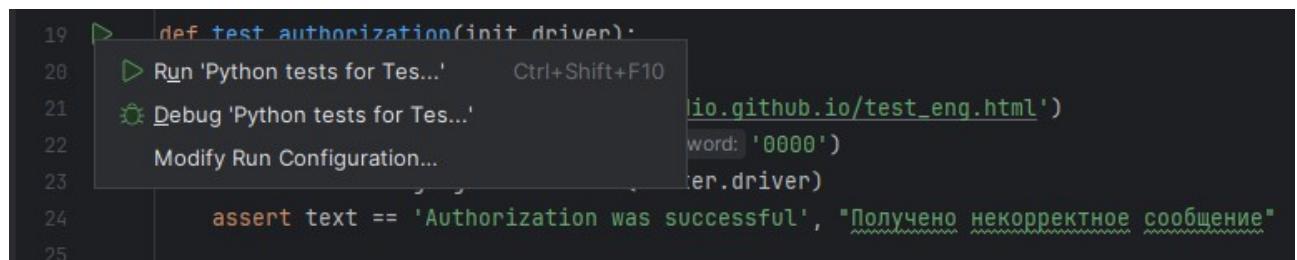
```
import pytest

from selenium import webdriver
from support.PageObjects.CommonPage import CommonPage
from support.StepObjects.CommonSteps import CommonSteps

@pytest.fixture(scope="session")
def init_driver():
    # Setup
    driver = webdriver.Chrome()
    driver.maximize_window()
    # Test
    yield driver
    # Teardown
    driver.close()
    driver.quit()

def test_authorization(init_driver):
    tester = CommonSteps(init_driver)
    tester.driver.get('https://somovstudio.github.io/test_eng.html')
    tester.sendForm( 'admin', '0000')
    text = CommonPage.getResultText(tester.driver)
    assert text == 'Authorization was successful', "Получено некорректное сообщение"
```

Автотест нужно запускать на функции test_authorization



Параметризация тестовых данных

Чтобы задать список данных для тестирования нужно использовать фикстуру
`@pytest.mark.parametrize`

В файле TestFixture.py описать автотест следующим образом

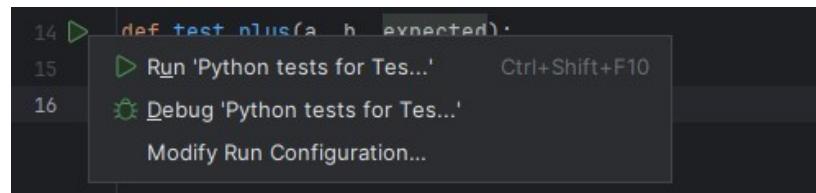
```
import pytest

def plus(a, b):
    return a + b

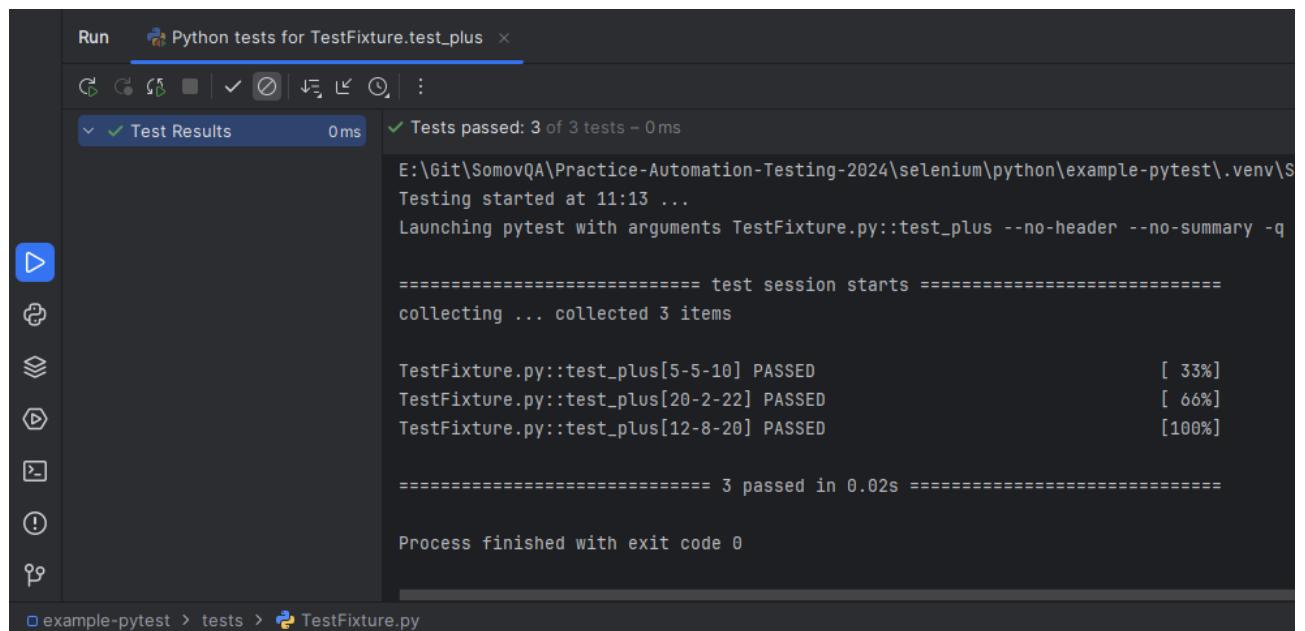
def minus(a, b):
    return a - b

@pytest.mark.parametrize("a, b, expected", [
    (5, 5, 10),
    (20, 2, 22),
    (12, 8, 20)
])
def test_plus(a, b, expected):
    actual = plus(a, b)
    assert expected == actual
```

Запуск автотеста



В результате тест будет запущен три раза под указанные параметры: a, b, expected



Полезные ссылки:

- Официальная документация Fixture
<https://docs.pytest.org/en/6.2.x/fixture.html>

Фреймворк Cucumber

Скачать и

Фреймворк *Selenide*

Скачать и

Фреймворк *Robot*

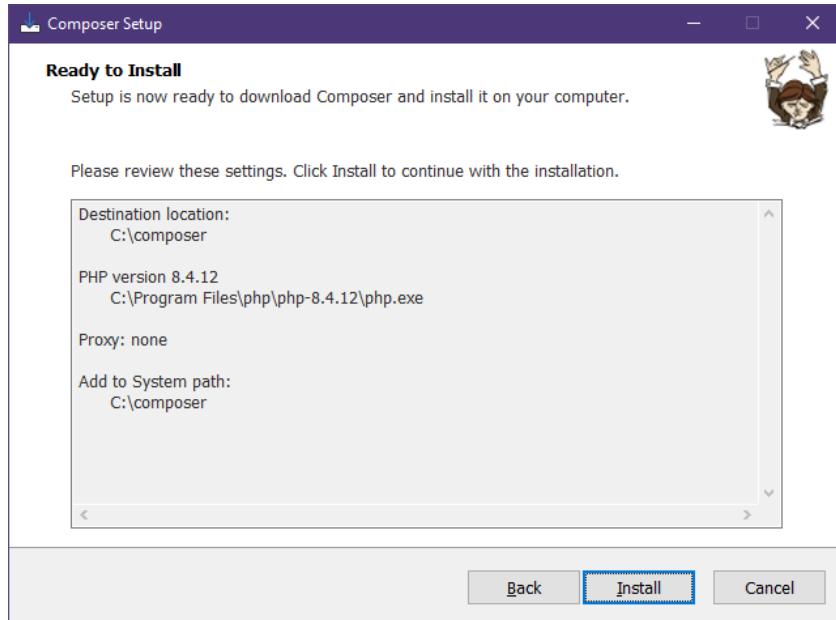
Скачать и

Фреймворк Codeception

Для работы с Codeception сначала необходимо скачать и установить [PHP](#) версии 8 с официального сайта.

Затем скачайте и установите [Composer](#)

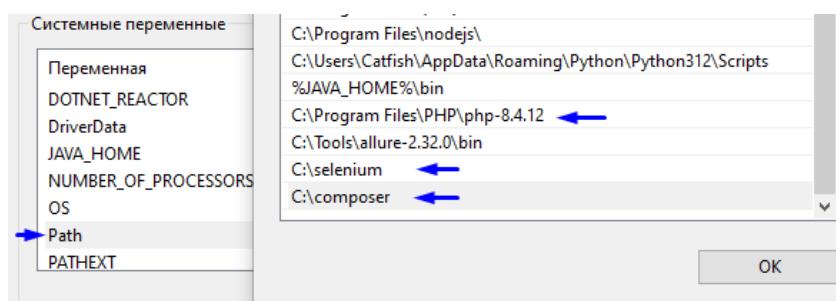
При установке включите Developer mode, выберите папку для установки (например C:\composer\), укажите путь к установленному PHP



После установки рекомендуется перезагрузить компьютер.

В переменной **Path** нужно прописать путь к папке C:\composer

(Панель управления > Система > Дополнительные параметры системы > Переменные среды)



Чтобы проверить установку Composer откройте командную строку и введите:

```
composer -V
```

```
Microsoft Windows [Version 10.0.19045.6332]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

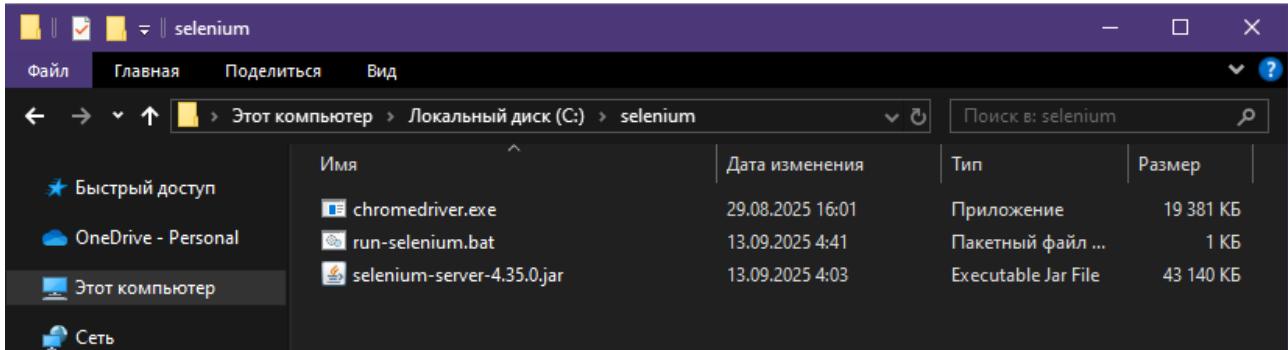
C:\Users\Catfish>composer -V
Composer version 2.8.11 2025-08-21 11:29:39
PHP version 8.4.12 (C:\Program Files\PHP\php-8.4.12\php.exe)
Run the "diagnose" command to get more detailed diagnostics output.

C:\Users\Catfish>
```

если всё было правильно сделано вы увидите версию Composer и PHP.

Следующее что нужно сделать так это подготовить Selenium Server (Grid) к работе. Как это сделать описано в главе «[Установка и запуск Selenium Grid](#)» данной книги. В папке C:\selenium\ запустите run-selenium.bat в котором вызывается запуск сервера:

```
cd C:\selenium  
java -jar selenium-server-4.24.0.jar standalone
```



```
C:\Windows\system32\cmd.exe  
C:\selenium>cd C:\selenium  
C:\selenium>java -jar selenium-server-4.35.0.jar standalone  
05:30:12.104 INFO [LoggingOptions.configureLogEncoding] - Using the system default encoding  
05:30:12.104 INFO [OpenTelemetryTracer.createTracer] - Using OpenTelemetry for tracing  
05:30:13.351 INFO [NodeOptions.getSessionFactories] - Detected 4 available processors  
05:30:13.353 INFO [NodeOptions.discoverDrivers] - Looking for existing drivers on the PATH.  
05:30:13.353 INFO [NodeOptions.discoverDrivers] - Add '--selenium-manager true' to the startup command to setup drivers automatically.  
05:30:13.538 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper msedgedriver version in offline mode  
05:30:13.636 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper geckodriver version in offline mode  
05:30:13.700 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper IEDriverServer version in offline mode  
05:30:13.719 INFO [NodeOptions.report] - Adding Firefox for {"browserName": "firefox", "platformName": "Windows 10"} 4 times  
05:30:13.720 INFO [NodeOptions.report] - Adding Chrome for {"browserName": "chrome", "platformName": "Windows 10"} 4 times  
05:30:13.721 INFO [NodeOptions.report] - Adding Edge for {"browserName": "MicrosoftEdge", "platformName": "Windows 10"} 4 times  
05:30:13.722 INFO [NodeOptions.report] - Adding Internet Explorer for {"browserName": "internet explorer", "platformName": "Windows 10"} 1 times  
05:30:13.787 INFO [Node.<init>] - Binding additional locator mechanisms: relative  
05:30:13.810 INFO [LocalGridModel.setAvailability] - Switching Node f58edb48-51eb-4a93-8122-514277b3dc0e (uri: http://192.168.132.1:4444) from DOWN to UP  
05:30:13.811 INFO [LocalNodeRegistry.add] - Added node f58edb48-51eb-4a93-8122-514277b3dc0e at http://192.168.132.1:4444. Health check every 120s  
05:30:14.147 INFO [Standalone.execute] - Started Selenium Standalone 4.35.0 (revision 1c58e5028b): http://192.168.132.1:4444
```

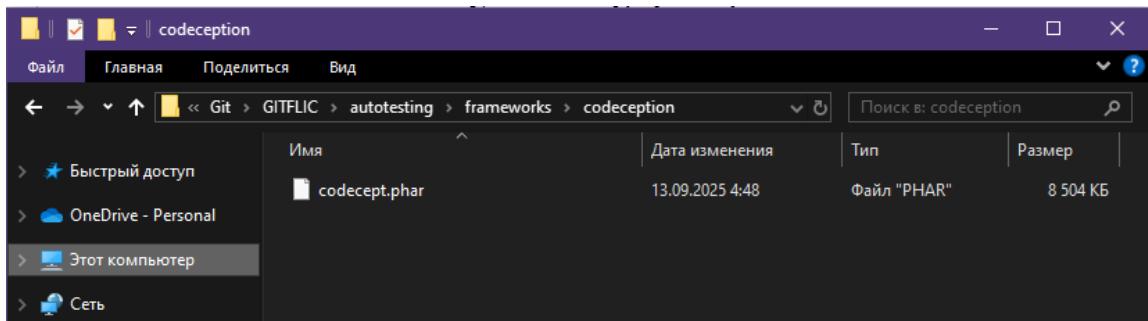
После того как сервер подготовлен и запущен можно приступать к установке Codeception.

Быстрая установка Codeception

Скачайте с официального сайта [Codeception](#) файл codeception.phar (версии 5)



и поместите файл codeception.phar в папку с будущим проектом.

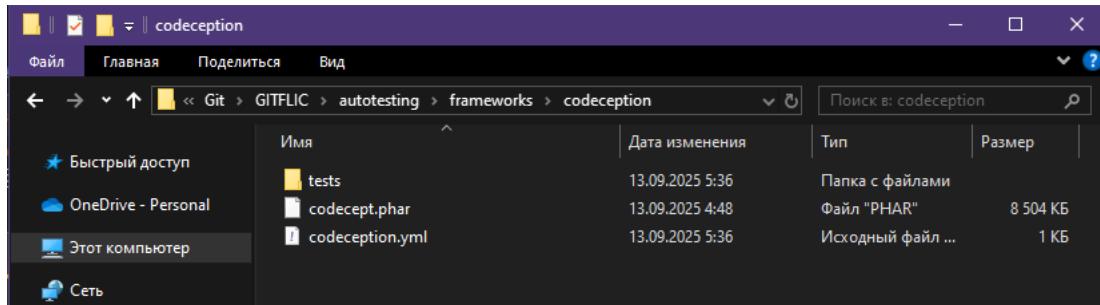


Чтобы создать пространство для тестов необходимо выполнить команду:

```
php codecept.phar bootstrap
```

A screenshot of a Windows Command Prompt window titled 'C:\Windows\System32\cmd.exe'. The command 'E:\Git\GITFLIC\autotesting\frameworks\codeception>php codecept.phar bootstrap' is run, followed by 'Bootstrapping Codeception'. The output shows the creation of configuration files like 'codeception.yml', 'tests/Unit/*.yml', 'tests/Functional/*.yml', and 'tests/Acceptance/*.yml'. A green box highlights the message 'Codeception is installed for acceptance, functional, and unit testing'. At the bottom, 'Next steps:' is listed with five numbered items: 1. Edit tests/acceptance.suite.yml, 2. Edit tests/functional.suite.yml, 3. Create your first acceptance tests using codecept g:cest acceptance First, 4. Write first test in tests/acceptance/FirstCest.php, and 5. Run tests using: codecept run.

в результате будет создан каталог tests со всей необходимой структурой и файлами



на этом установка Codeception завершена и можно приступить к созданию автотеста.

Создание автотеста.

Перед созданием автотеста необходимо отредактировать файл настроек приемочных автотестов Acceptance.suite.yml в папке tests

Имя	Дата изменения	Тип	Размер
_output	13.09.2025 5:36	Папка с файлами	
Acceptance	13.09.2025 5:36	Папка с файлами	
Functional	13.09.2025 5:36	Папка с файлами	
Support	13.09.2025 5:36	Папка с файлами	
Unit	13.09.2025 5:36	Папка с файлами	
Acceptance.suite.yml	13.09.2025 5:36	Исходный файл ...	1 КБ
Functional.suite.yml	13.09.2025 5:36	Исходный файл ...	1 КБ
Unit.suite.yml	13.09.2025 5:36	Исходный файл ...	1 КБ

В файле Acceptance.suite.yml включим поддержку WebDriver и опишем его config.

```
actor: AcceptanceTester
modules:
    enabled:
        - WebDriver:
            url: https://somovstudio.github.io
            browser: chrome
            window_size: 1440x900
        # - PhpBrowser:
        #     url: http://localhost/myapp
        # - \Helper\Acceptance
    config:
        WebDriver:
            capabilities:
                chromeOptions:
                    args: ["--ignore-certificate-errors"]

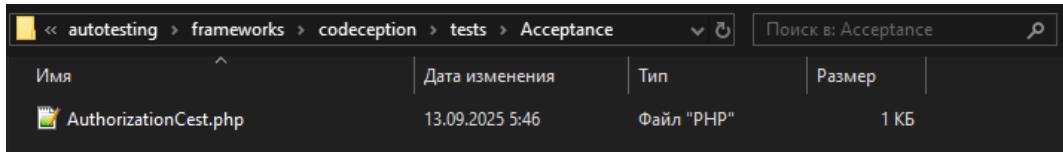
# Add Codeception\Step\Retry trait to AcceptanceTester to enable retries
step_decorators:
    - Codeception\Step\ConditionalAssertion
    - Codeception\Step\TryTo
    - Codeception\Step\Retry
```

Чтобы создать автотест Authorization нужно выполнить команду

```
php codecept.phar g:cest Acceptance Authorization
```

```
E:\Git\GITFLIC\autotesting\frameworks\codeception>php codecept.phar g:cest Acceptance Authorization
Test was created in E:\Git\GITFLIC\autotesting\frameworks\codeception\tests\Acceptance\AuthorizationCest.php
E:\Git\GITFLIC\autotesting\frameworks\codeception>
```

В папке будет создан файл автотест AuthorizationCest.php



Опишите автотест в файле AuthorizationCest.php для проверки авторизации следующим образом:

```
<?php
namespace Tests\Acceptance;
use Tests\Support\AcceptanceTester;
use PHPUnit\Framework\Assert as PHPAssert;

class AuthorizationCest
{
    public function _before(AcceptanceTester $I)
    {
        $I->wantTo('Проверка авторизации');
        $I->maximizeWindow();
    }

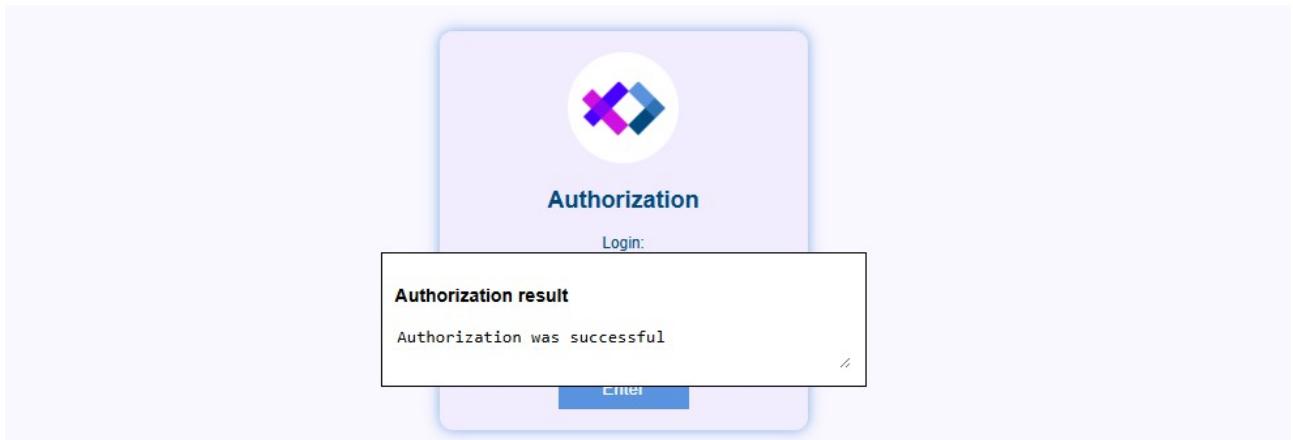
    public function tryToTest(AcceptanceTester $I)
    {
        $I->amOnPage('/test_eng.html');
        $I->wait(1);
        $I->fillField('login', 'admin');
        $I->wait(1);
        $I->fillField('pass', '0000');
        $I->wait(1);
        $I->click('Enter');
        $I->wait(1);
        $I->see('Authorization result');
        $text = $I->grabValueFrom("//textarea[@id='textarea']");
        PHPAssert::assertEquals('Authorization was successful', $text);
    }
}
```

A screenshot of a code editor displaying the 'AuthorizationCest.php' file. The code is identical to the one above, but the code editor uses syntax highlighting to distinguish between different parts of the code. The background is dark, and the code is in a light color. Braces and brackets are highlighted with vertical dotted lines, and the class and method names are in bold. The file number 1 through 32 is visible on the left side of the editor.

Чтобы запустить автотест необходимо выполнить команду:

```
php codecept.phar run Acceptance AuthorizationCest --steps
```

В результате в консоли будет отображен процесс тестирования и его результат.



```
C:\Windows\System32\cmd.exe - X

E:\Git\GITFLIC\autotesting\frameworks\codeception>php codecept.phar run Acceptance AuthorizationCest --steps
Codeception PHP Testing Framework v5.1.2 https://stand-with-ukraine.pp.ua

Tests.Acceptance Tests (1) -----
AuthorizationCest: Try to test
Signature: Tests\Acceptance\AuthorizationCest:tryToTest
Test: tests\Acceptance\AuthorizationCest.php:tryToTest
Scenario --
I maximize window
I am on page "/test_eng.html"
I wait 1
I fill field "login","admin"
I wait 1
I fill field "pass","0000"
I wait 1
I click "Enter"
I wait 1
I see "Authorization result"
I grab value from "//textarea[@id='textarea']"
PASSED

-----
Time: 00:08.305, Memory: 22.00 MB
OK (1 test, 2 assertions)

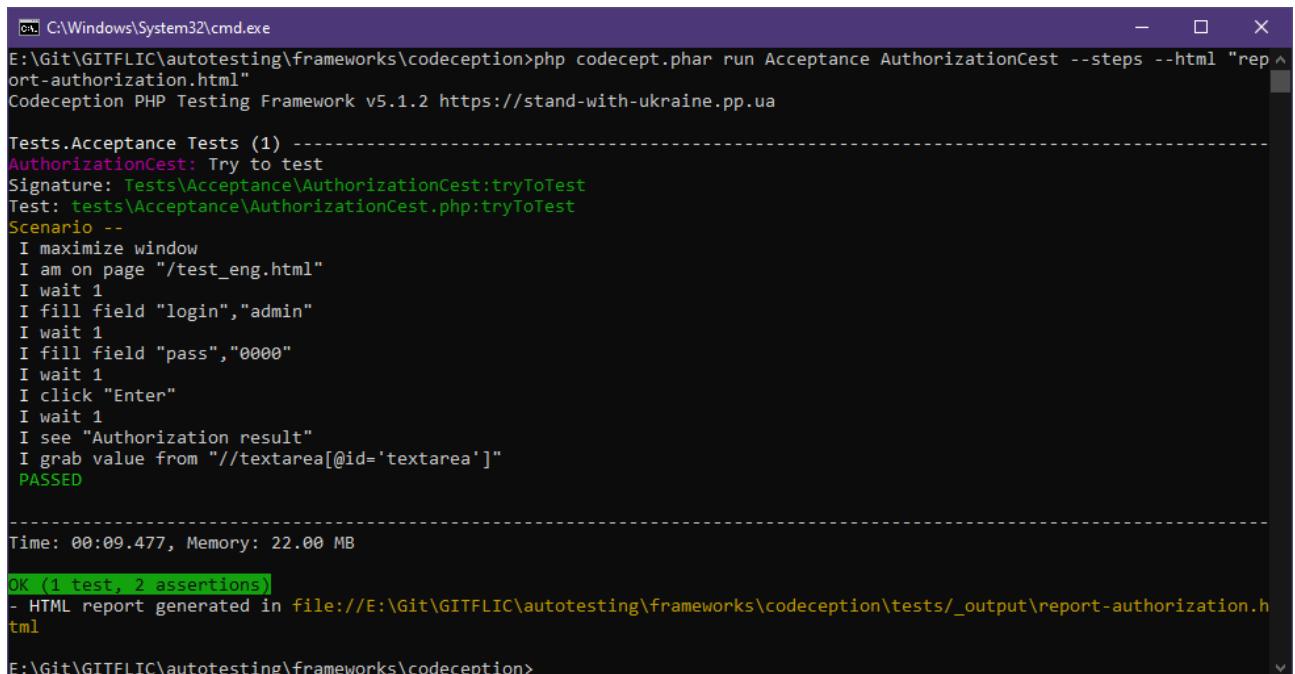
E:\Git\GITFLIC\autotesting\frameworks\codeception>
```

Отчет.

Чтобы получить отчет необходимо при запуске указать параметр --html в команде:

```
php codecept.phar run Acceptance AuthorizationCest --steps --html "report-authorization.html"
```

После выполнения автотест по адресу \tests_output\ будет сформирован отчет в файле report-authorization.html в формате html.



The screenshot shows a Windows Command Prompt window titled 'C:\Windows\System32\cmd.exe'. The command executed is 'php codecept.phar run Acceptance AuthorizationCest --steps --html "report-authorization.html"'. The output shows the test steps and their results. The test passes with a duration of 00:09.477 and memory usage of 22.00 MB. An HTML report is generated at the end.

```
E:\Git\GITFLIC\autotesting\frameworks\codeception>php codecept.phar run Acceptance AuthorizationCest --steps --html "report-authorization.html"
Codeception PHP Testing Framework v5.1.2 https://stand-with-ukraine.pp.ua

Tests.Acceptance Tests (1) --
AuthorizationCest: Try to test
Signature: Tests\Acceptance\AuthorizationCest:tryToTest
Test: tests\Acceptance\AuthorizationCest.php:tryToTest
Scenario --
I maximize window
I am on page "/test_eng.html"
I wait 1
I fill field "login", "admin"
I wait 1
I fill field "pass", "0000"
I wait 1
I click "Enter"
I wait 1
I see "Authorization result"
I grab value from "//textarea[@id='textarea']"
PASSED

Time: 00:09.477, Memory: 22.00 MB

OK (1 test, 2 assertions)
- HTML report generated in file:///E:/Git/GITFLIC/autotesting/frameworks/codeception/tests/_output/report-authorization.html

E:\Git\GITFLIC\autotesting\frameworks\codeception>
```

Codeception Results OK (00:09.437)

Tests.Acceptance Tests

- AuthorizationCest » Try to test 8.16s

```
I maximize window
I am on page "/test_eng.html"
I wait 1
I fill field "login", "admin"
I wait 1
I fill field "pass", "0000"
I wait 1
I click "Enter"
I wait 1
I see "Authorization result"
I grab value from "//textarea[@id='textarea']"
```

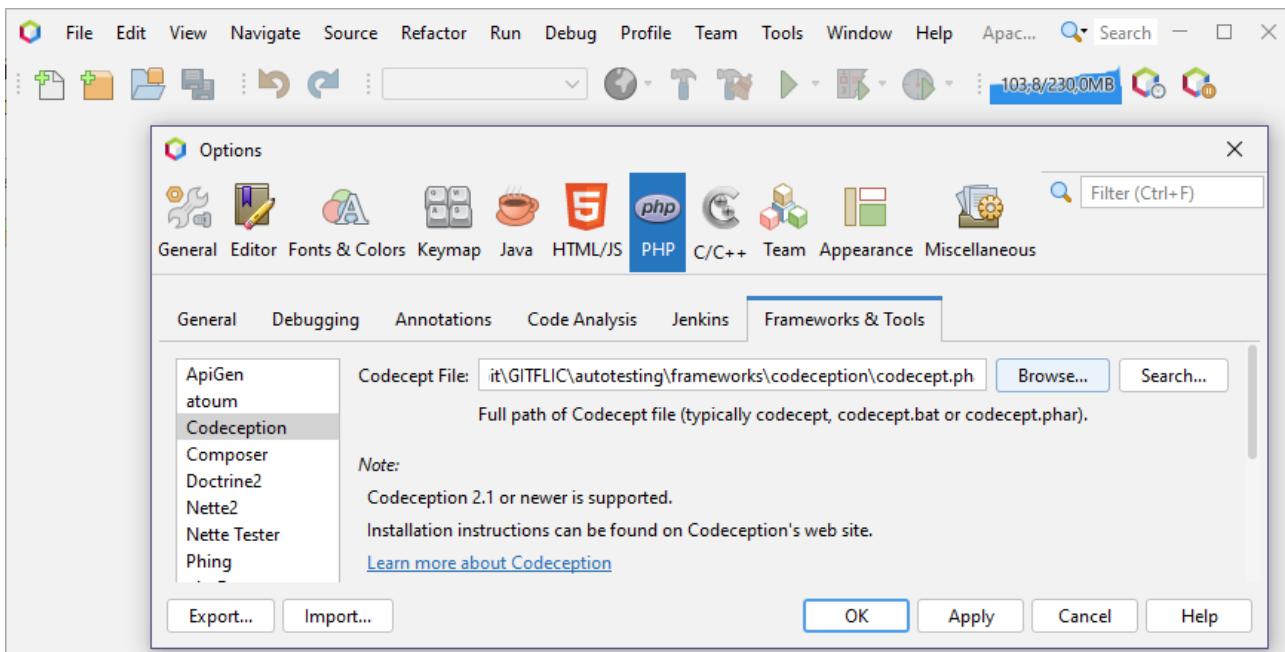
Summary

Successful scenarios:	1
Failed scenarios:	0
Skipped scenarios:	0
Incomplete scenarios:	0
Useless scenarios:	0

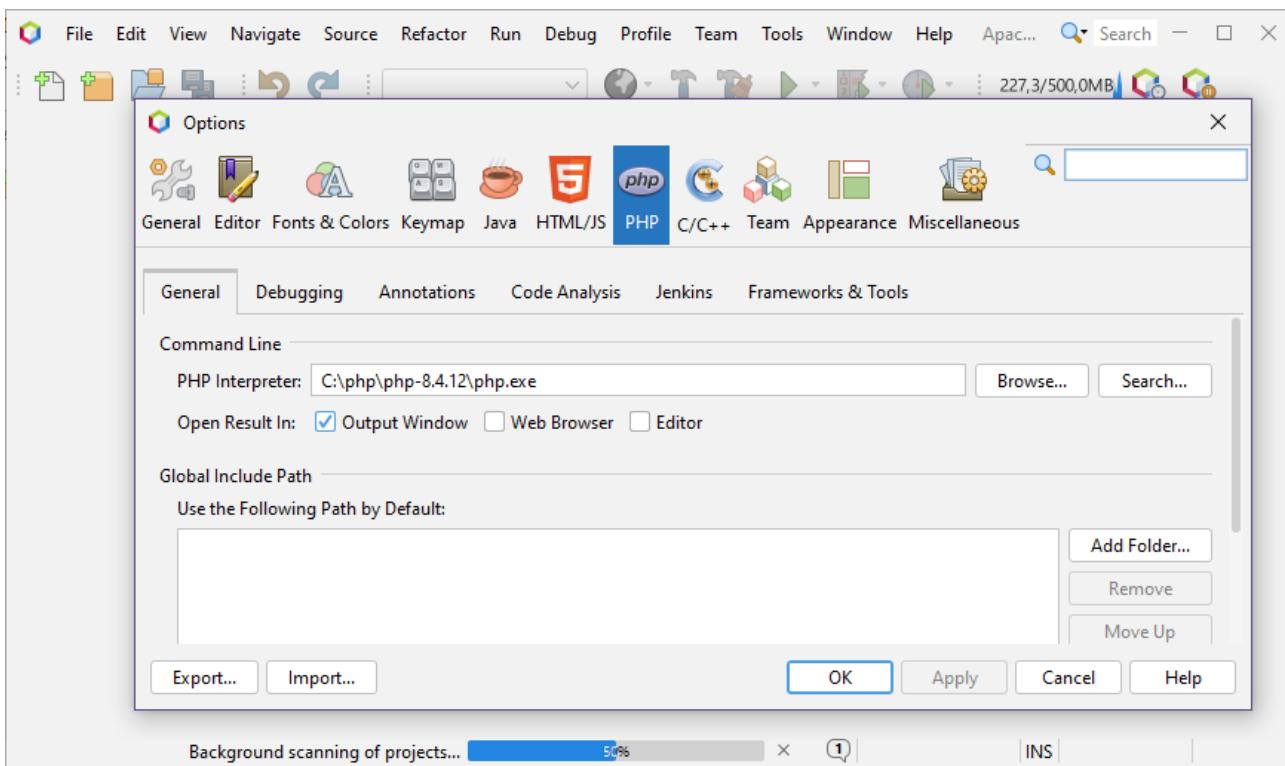
Редактор кода IDE.

Для работы с Codeception очень удобно использовать бесплатный редактор Apache NetBeans IDE который можно скачать с официального сайта: <https://netbeans.apache.org/front/main/>

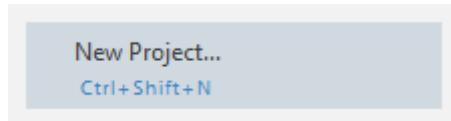
Скачайте и установите Apache NetBeans IDE. После запуска перейдите в меню Tools → Options. Во вкладке PHP найдите раздел Codeception и укажите адрес к файлу codecept.phar и нажимаем Apply (OK)



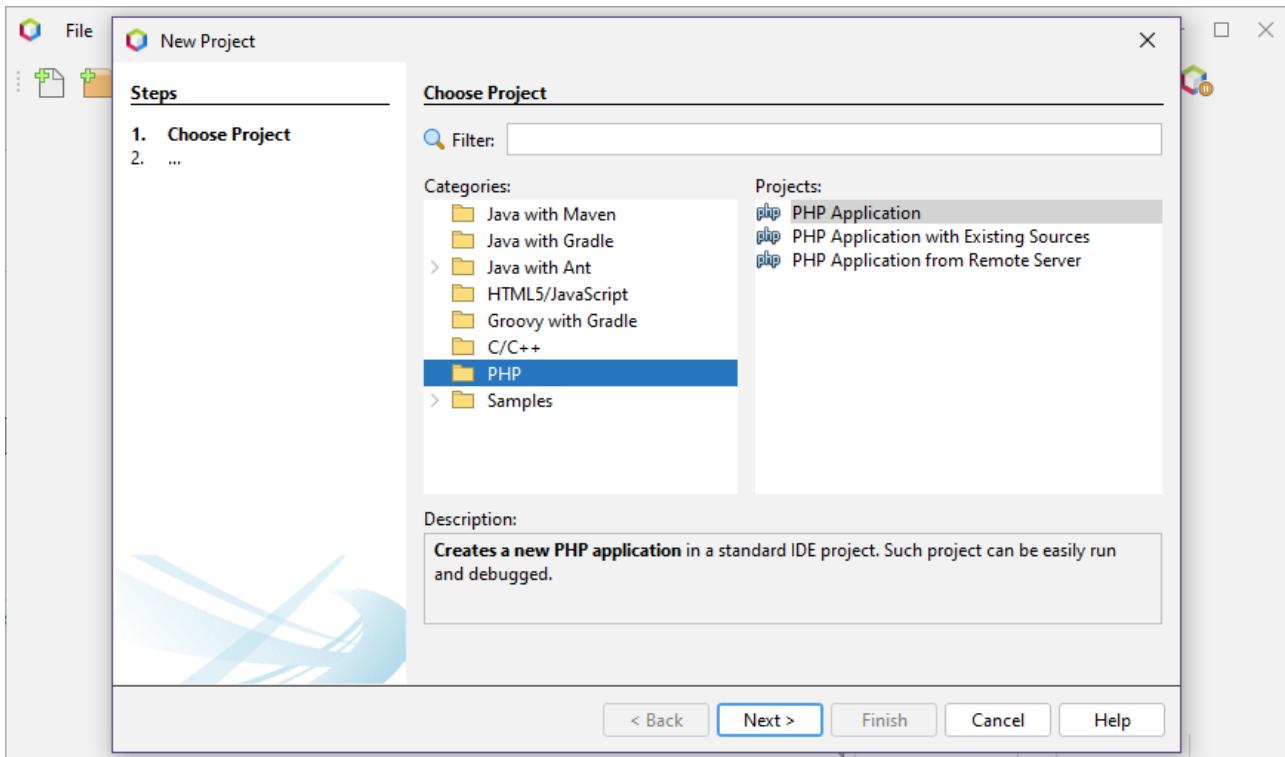
Дальше будет предложено указать путь к установленному PHP



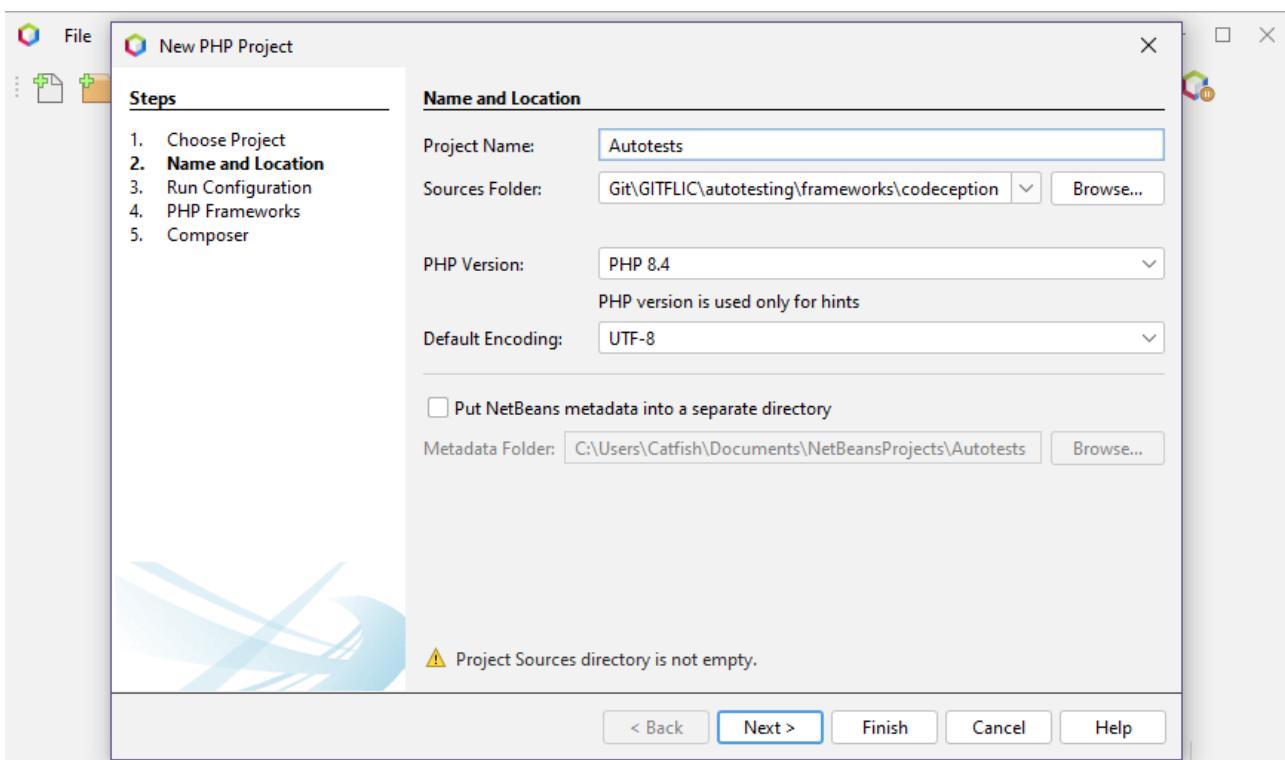
После всех настроек необходимо создать новый проект.



Выберите тип проекта PHP и нажмите Next



Введите путь к папке с автотестами на Codeception нажмите Finish (или Next)



Теперь вести разработку автотестов будет гораздо проще и удобнее поскольку при вводе методов вам будет показан список доступных методов Codeception.

The screenshot shows the PhpStorm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Run, along with memory usage information (574.7 / 734.0MB).
- Project Structure:** Shows the project structure under "Autotests [master]". The "Source Files" folder contains "tests" which has "Acceptance" and "AuthorizationCest.php". "AuthorizationCest.php" is the active file.
- Code Editor:** Displays the PHP code for "AuthorizationCest.php". The code defines a class "AuthorizationCest" with two methods: "_before" and "tryToTest". The "tryToTest" method uses the \$I variable to perform various actions like navigating to a page, filling fields, and asserting results.
- Navigator:** Shows the navigation tree with nodes for "Tests\Acceptance", "Tests\Support\AcceptanceTester", and "AuthorizationCest" with its methods: "_before(AcceptanceTester \$I)" and "tryToTest(AcceptanceTester \$I)".
- Code Completion:** A tooltip is shown over the "\$I->" part of the code, listing available methods from the "AcceptanceTester" class. The "amOnPage(\$page)" method is highlighted.

```

<?php
namespace Tests\Acceptance;

use Tests\Support\AcceptanceTester;
use PHPUnit\Framework\Assert as PHPAssert;

class AuthorizationCest
{
    public function _before(AcceptanceTester $I)
    {
        $I->wantTo('Проверка авторизации');
        $I->maximizeWindow();
    }

    // tests
    public function tryToTest(AcceptanceTester $I)
    {
        $I->amOnPage('/test_eng.html');
        $I->wait(1);
        $I->fillField('login', 'admin');
        $I->wait(1);
        $I->fillField('pass', '0000');
        $I->wait(1);
        $I->click('Enter');
        $I->wait(1);
        $I->see('Authorization result');
        $text = $I->grabValueFrom("//textarea[@id='textarea']");
        PHPAssert::assertEquals('Authorization was successful', $text);
    }
}

```

Применение Helper и паттернов PageObject, StepObject.

В Codeception можно создать и использовать шаблоны кода которые упрощают разработку автотестов. Для начала познакомимся с Helper.

Чтобы создать Helper скрипт нужно выполнить команду

```
php codecept.phar generate:helper AcceptanceHelper
```

в результате по адресу \tests\Support\Helper\ будет создан файл AcceptanceHelper.php в котором необходимо описать метод и константу следующим образом:

```
<?php

namespace Tests\Support\Helper;
use Codeception\Module;
use PHPUnit\Framework\Assert as PHPAssert;

class AcceptanceHelper extends Module
{
    const URL_TEST_PAGE = 'https://somovstudio.github.io/test_eng.html';

    public function checkResult($text)
    {
        PHPAssert::assertEquals('Authorization was successful', $text);
    }
}
```

Теперь создадим PageObject скрипт с помощью команды:

```
php codecept.phar generate:pageobject Acceptance LoginPage
```

в результате по адресу \tests\Support\Page\Acceptance\ будет создан файл LoginPage.php в котором необходимо описать метод и переменные следующим образом:

```
<?php
declare(strict_types=1);
namespace Tests\Support\Page\Acceptance;

class LoginPage
{
    protected $acceptanceTester;

    public $usernameField = 'login';
    public $passwordField = 'pass';
    public $loginButton = 'Enter';
    public $textMessage = "//textarea[@id='textarea']";

    public function __construct(\Tests\Support\AcceptanceTester $I)
    {
        $this->acceptanceTester = $I;
    }

    public function login($username, $password)
    {
        $this->acceptanceTester->fillField($this->usernameField, $username);
        $this->acceptanceTester->wait(1);
        $this->acceptanceTester->fillField($this->passwordField, $password);
        $this->acceptanceTester->wait(1);
        $this->acceptanceTester->click($this->loginButton);
        $this->acceptanceTester->wait(1);
        return $this;
    }
}
```

И создадим StepObject скрипт с помощью команды:

```
php codecept.phar generate:stepobject Acceptance UserSteps
```

в результате по адресу \tests\Support\Step\Acceptance\ будет создан файл UserSteps.php в котором необходимо описать метод авторизации следующим образом:

```
<?php
declare(strict_types=1);
namespace Tests\Support\Step\Acceptance;
use Tests\Support\Page\Acceptance\LoginPage;

class UserSteps extends \Tests\Support\AcceptanceTester
{
    public function login($username, $password)
    {
        $I = $this;
        $LoginPage = new LoginPage($I);
        $LoginPage->login($username, $password);
        return $this;
    }
}
```

Остается создать автотест который будет использовал код из шаблонов описанных ранее.

Файл автотesta testAuthorizationCest.php

```
<?php
namespace Tests\Acceptance;
use Tests\Support\AcceptanceTester;
use Tests\Support\Step\Acceptance\UserSteps;

class testAuthorizationCest
{
    public function _before(AcceptanceTester $I)
    {
        $I->wantTo('Проверка авторизации');
        $I->maximizeWindow();
    }

    // tests
    public function tryToTest(UserSteps $I)
    {
        $I->amOnUrl(\Tests\Support\Helper\AcceptanceHelper::URL_TEST_PAGE); // Helper
        $I->login('admin', '0000'); // StepObject, PageObject
        $I->see('Authorization result');
        $text = $I->grabValueFrom("//textarea[@id='textarea']");
        $I->checkResult($text); // Helper
    }
}
```

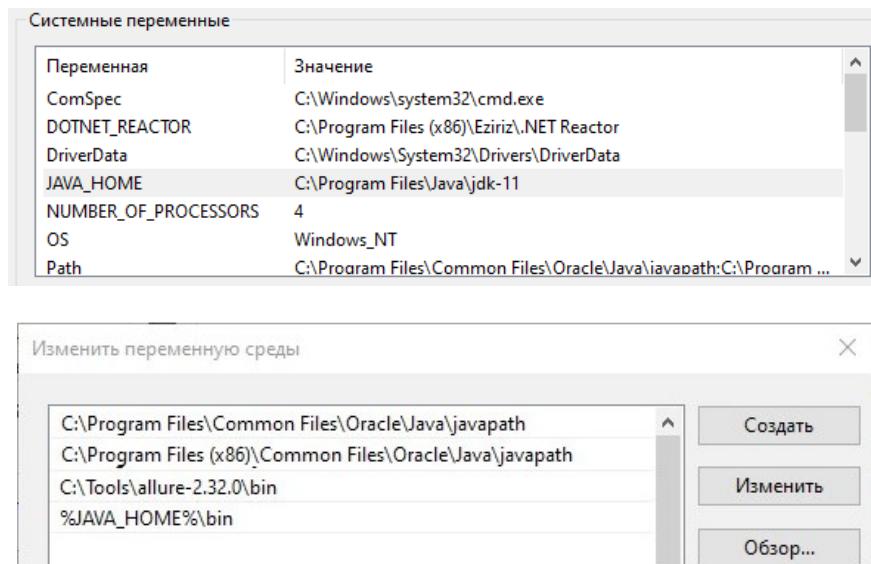
Полезные ссылки

- Официальный сайт PHP <https://www.php.net/downloads.php>
- Официальный сайт Composer <https://getcomposer.org/>
- Официальный сайт Selenium <https://www.selenium.dev/downloads/>
- Сайт Webdriver для Chrome <https://googlechromelabs.github.io/chrome-for-testing/>
- Официальный сайт Codeception <https://codeception.com/install>
- Описание всех методов WebDriver <https://codeception.com/docs/modules/WebDriver>
- Официальный сайт NetBeans <https://netbeans.apache.org/front/main/>

Отчет Allure Report

Установить Allure Report из архива для Windows

Убедитесь, что установлена Java версии 8 или выше, а ее каталог указан в **JAVA_HOME** переменной среды.



Чтобы узнать версию Java нужно выполнить команду:

```
java -version
```

```
Командная строка
Microsoft Windows [Version 10.0.19045.5247]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>java -version
java version "11.0.21" 2023-10-17 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.21+9-LTS-193)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.21+9-LTS-193, mixed mode)

C:\Users\Catfish>
```

Перейдите к последней версии Allure Report на GitHub
<https://github.com/allure-framework/allure2/releases/tag/2.32.0>

и загрузите архив allure-* .zip или allure-* .tgz.

▼ Assets 6

[allure-2.32.0.tgz](#)

[allure-2.32.0.zip](#)

Распакуйте архив в каталог по вашему выбору.

Локальный диск (C:) > Tools > allure-2.32.0			
Имя	Дата изменения	Тип	Размер
bin	29.10.2024 10:06	Папка с файлами	
config	29.10.2024 10:06	Папка с файлами	
lib	29.10.2024 10:06	Папка с файлами	
plugins	29.10.2024 10:06	Папка с файлами	

Запомните путь к его **bin** подкаталогу. (в моем случае <C:\Tools\allure-2.32.0\bin>)
Этот путь понадобится вам на следующем шаге.

Убедитесь, что allure команда разрешает файл allure из вашего установочного каталога.

Это можно сделать разными способами, например, через Панель управления.

1. Нажмите Win+R и введите команду: [sysdm.cpl](#) чтобы открыть инструмент «Свойства системы».
2. На вкладке Дополнительно нажмите Переменные среды .
3. В списке переменных пользователя или системных переменных дважды щелкните [Path](#) переменную, чтобы открыть диалоговое окно редактирования. Обратите внимание, что редактирование системной переменной требует прав администратора и влияет на всех пользователей компьютера.
4. В диалоговом окне «Изменить переменную среды» нажмите кнопку «Создать», чтобы добавить новую запись строки в список путей. В новой строке укажите полный путь к подкаталогу **bin** из предыдущего шага, например: <C:\Tools\allure-2.32.0\bin>.



5. Если в списке содержится путь к ранее установленной версии Allure, удалите ее.
6. Нажмите «OK», чтобы сохранить изменения.

В консоли выполните команду, чтобы проверить установленную версию allure:

```
allure --version
```

```
Командная строка
Microsoft Windows [Version 10.0.19045.5247]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

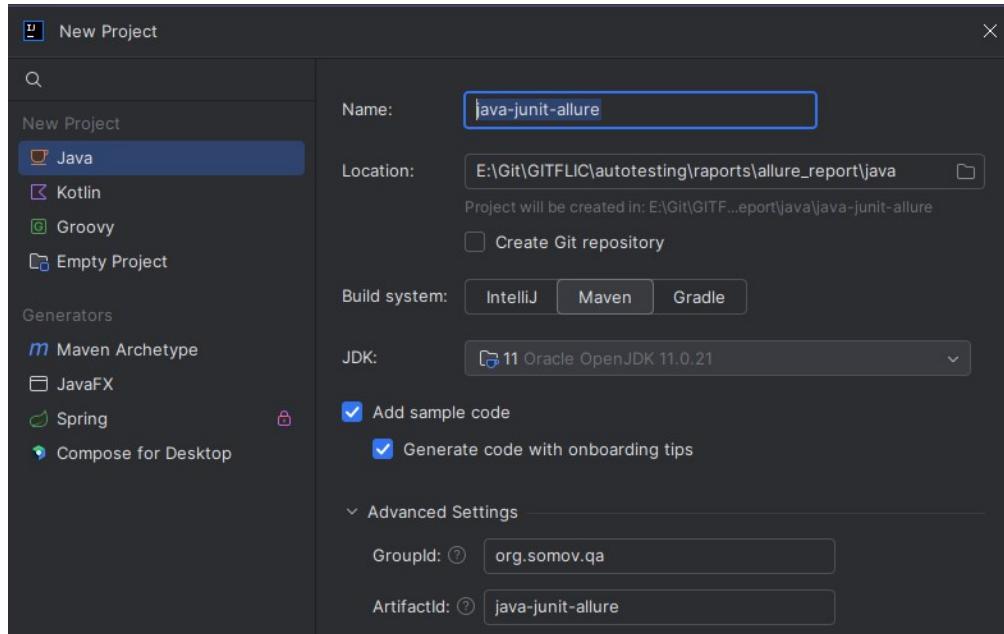
C:\Users\Catfish>allure --version
2.32.0
C:\Users\Catfish>
```

Полезные ссылки:

- Официальная страница <https://allurereport.org/>
- Официальная документация <https://allurereport.org/docs/>
- Быстрый старт PyTest <https://pypi.org/project/allure-pytest/2.13.5/>

Практика применения Allure Report (Java)

Чтобы использовать Allure Report в проектах Java нужно выполнить следующие действия.
Создать Java ([JUnit5](#)) проект под именем java-junit-allure



Подключить библиотеки JUnit, Selenium и Allure в файле pom.xml

ссылки на библиотеки:

- Selenium: <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
- JUnit: <https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api>
- Allure: <https://mvnrepository.com/artifact/io.qameta.allure/allure-junit5>

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.34.0</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.11.3</version>
        <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/io.qameta.allure/allure-junit5 -->
    <dependency>
        <groupId>io.qameta.allure</groupId>
        <artifactId>allure-junit5</artifactId>
        <version>2.29.1</version>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-simple</artifactId>
        <version>2.0.16</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Создать простой автотест TestAuthorization.java с поддержкой Allure аннотаций

```
package tests;

import io.qameta.allure.Description;
import io.qameta.allure.Issue;
import io.qameta.allure.Link;
import io.qameta.allure.Owner;
import io.qameta.allure.Severity;
import io.qameta.allure.TmsLink;

import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Tag;
import org.junit.jupiter.api.Test;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.Objects;

public class TestAuthorization {
    public static WebDriver driver;

    @BeforeAll
    public static void before(){
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

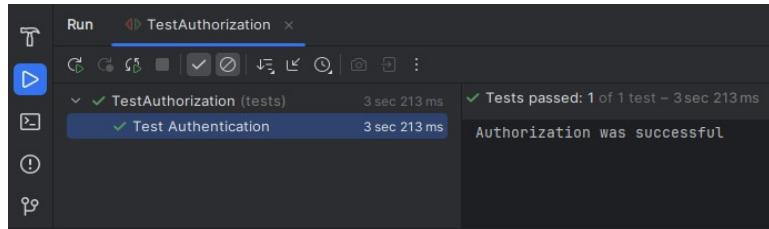
    @Test
    @DisplayName("Test Authentication")
    @Description("Проверка авторизации на сайте.")
    @Severity(io.qameta.allure.SeverityLevel.CRITICAL)
    @Owner("Tester QA")
    @Link(name = "Website", url = "https://somovstudio.github.io/")
    @Issue("ISSUE-1")
    @TmsLink("TEST CASE-1")
    public void testAuthorization() {
        driver.get("https://somovstudio.github.io/test_eng.html");
        driver.findElement(By.name("login")).sendKeys("admin");
        driver.findElement(By.name("pass")).sendKeys("0000");
        driver.findElement(By.id("buttonLogin")).click();

        WebElement element = driver.findElement(By.id("result"));
        Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());

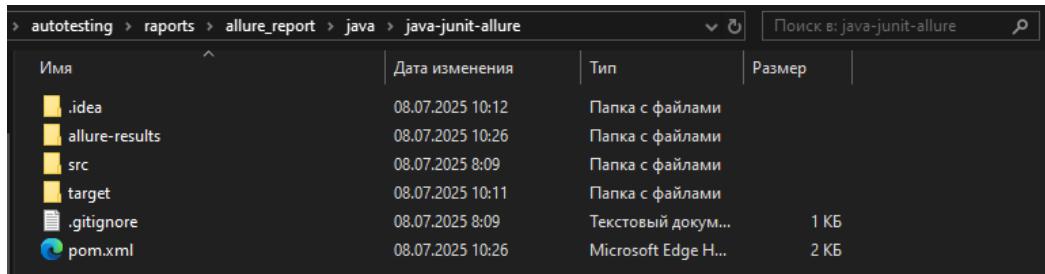
        String text = driver.findElement(By.id("textarea")).getAttribute("value");
        System.out.println(text);
        Assertions.assertEquals(text, "Authorization was successful");
    }

    @AfterAll
    public static void after(){
        driver.close();
        driver.quit();
    }
}
```

Выполните автотест запустив его в редакторе



Перейдите в папку проекта в которой должен был создаться каталог allure-results

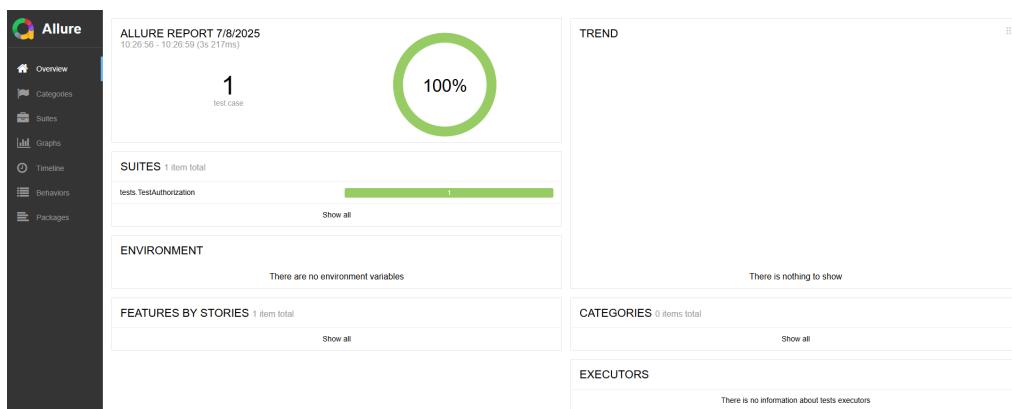


Откройте консоль и выполните команду:

```
allure serve allure-results
```

A screenshot of a Windows Command Prompt window. The title bar says 'C:\Windows\System32\cmd.exe - allure serve allure-results'. The command 'allure serve allure-results' is entered and executed. The output shows the report is generating to a temporary directory and successfully starting a web server at 'http://192.168.132.1:61384/'.

Эта команда сформирует отчет и откроет его в браузере



Полезные ссылки:

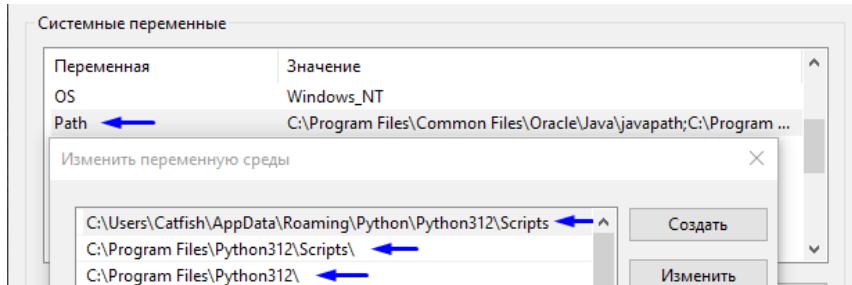
- Getting started with Allure JUnit 5 <https://allurereport.org/docs/junit5/>
- JUnit 5 parameterization <https://allurereport.org/docs/guides/junit5-parametrization/>
- Integrating screenshots and other attachments in Allure Report with Selenium and JUnit 5 <https://allurereport.org/docs/guides/junit5-selenium-screenshots/>

Практика применения Allure Report PyTest (Python)

Чтобы использовать Allure Report в проектах Python нужно выполнить следующие действия.
Установить и настроить [PyTest](#)

```
pip install pytest
pip3 install pytest
python.exe -m pip install --upgrade pip
```

Добавить адрес C:\Users\Имя пользователя\AppData\Roaming\Python\Python312\Scripts
в системную переменную среды Path



Для проверки выполнить команду

```
pip3 show pytest
pytest --version
```

Установите адаптер Allure Pytest.

```
pip install allure-pytest
```

The screenshot shows a Windows Command Prompt window titled 'Командная строка'. The command 'pip install allure-pytest' was run, and the output shows the package being downloaded and installed successfully. The terminal window has a dark theme.

```
C:\Users\Catfish>pip install allure-pytest
Defaulting to user installation because normal site-packages is not writeable
Collecting allure-pytest
  Downloading allure_pytest-2.14.3-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: pytest>=4.5.0 in c:\users\catfish\appdata\roaming\python\python312\site-packages (from allure-pytest) (8.3.3)
Collecting allure-python-commons==2.14.3 (from allure-pytest)
  Downloading allure_python_commons-2.14.3-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: attrs>=16.0.0 in c:\users\catfish\appdata\roaming\python\python312\site-packages (from allure-python-commons==2.14.3->allure-pytest) (24.2.0)
Requirement already satisfied: pluggy>=0.4.0 in c:\users\catfish\appdata\roaming\python\python312\site-packages (from allure-python-commons==2.14.3->allure-pytest) (1.5.0)
Requirement already satisfied: iniconfig in c:\users\catfish\appdata\roaming\python\python312\site-packages (from pytest>=4.5.0->allure-pytest) (2.0.0)
Requirement already satisfied: packaging in c:\users\catfish\appdata\roaming\python\python312\site-packages (from pytest>=4.5.0->allure-pytest) (24.2)
Requirement already satisfied: colorama in c:\users\catfish\appdata\roaming\python\python312\site-packages (from pytest>=4.5.0->allure-pytest) (0.4.6)
  Downloading allure_pytest-2.14.3-py3-none-any.whl (11 kB)
  Downloading allure_python_commons-2.14.3-py3-none-any.whl (16 kB)
Installing collected packages: allure-python-commons, allure-pytest
Successfully installed allure-pytest-2.14.3 allure-python-commons-2.14.3
C:\Users\Catfish>
```

Создать проект python-pytest-allure с одним простым автотестом

The screenshot shows the PyCharm IDE interface. On the left is the Project view, which includes a .venv folder, a tests directory containing __init__.py and TestAuthorization.py, and External Libraries and Scratches and Consoles sections. The main window displays the code for TestAuthorization.py. The code imports pytest and selenium, sets up a Chrome driver, navigates to a login page, inputs 'admin' for login and '0000' for password, clicks the login button, waits for the result, and prints the message from a text area. It also contains an assert statement and cleanup code. The file ends with a conditional block to run pytest.main if it's the main module.

```
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

def test_authorization():
    driver = webdriver.Chrome()
    driver.maximize_window()
    driver.get('https://somovstudio.github.io/test_eng.html')
    driver.find_element(By.NAME, value: 'login').send_keys('admin')
    driver.find_element(By.NAME, value: 'pass').send_keys('0000')
    driver.find_element(By.ID, value: 'buttonLogin').click()
    element = driver.find_element(By.ID, value: 'result')
    wait = WebDriverWait(driver, timeout=5)
    wait.until(lambda d: element.is_displayed())
    text = driver.find_element(By.ID, value: 'textarea').get_property('value')
    print("Get message: " + text)

    assert text == 'Authorization was successful', "Получено некорректное сообщение"

    driver.close()
    driver.quit()

if __name__ == '__main__':
    pytest.main(["-s", "TestAuthorization.py"])
```

В файле TestAuthorization.py описать автотест следующим образом

```
import pytest

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

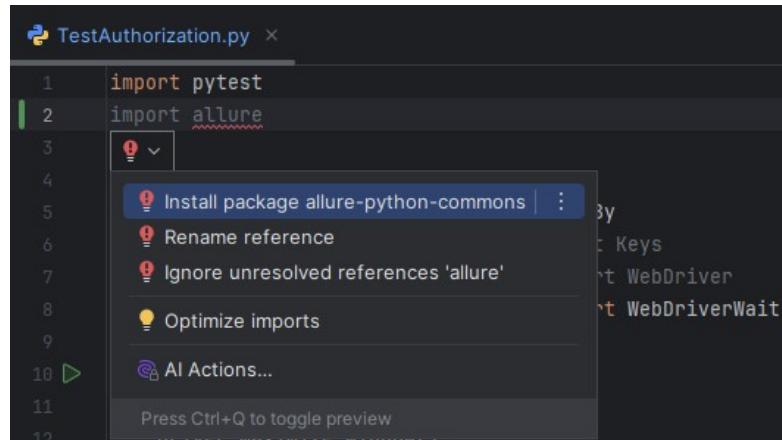
def test_authorization():
    driver = webdriver.Chrome()
    driver.maximize_window()
    driver.get('https://somovstudio.github.io/test_eng.html')
    driver.find_element(By.NAME, 'login').send_keys('admin')
    driver.find_element(By.NAME, 'pass').send_keys('0000')
    driver.find_element(By.ID, 'buttonLogin').click()
    element = driver.find_element(By.ID, 'result')
    wait = WebDriverWait(driver, timeout=5)
    wait.until(lambda d: element.is_displayed())
    text = driver.find_element(By.ID, 'textarea').get_property('value')
    print("Get message: " + text)

    assert text == 'Authorization was successful', "Получено некорректное сообщение"

    driver.close()
    driver.quit()

if __name__ == '__main__':
    pytest.main(["-s", "TestAuthorization.py"])
```

Импортировать Allure в проект и выполнить установку пакета



Изменить автотест TestAuthorization под отчет Allure следующим образом

```
import pytest
import allure

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

@allure.title("Test Authorization")
@allure.description("Проверка авторизации на сайте.\n\nВыполняется проверка авторизации на тестовой странице.")
@allure.tag("Test", "Authorization")
@allure.severity(allure.severity_level.CRITICAL)
@allure.label("owner", "Tester QA")
@allure.link("https://somovstudio.github.io/", name="Website")
@allure.issue("ISSUE-1")
@allure.testcase("TEST CASE-1")
def test_authorization():
    driver = webdriver.Chrome()
    driver.maximize_window()

    steps = Steps()
    steps.step1(driver)
    steps.step2(driver)

    driver.close()
    driver.quit()

class Steps:

    @allure.step("Step 1")
    def step1(self, driver):
        driver.get('https://somovstudio.github.io/test_eng.html')
        driver.find_element(By.NAME, 'login').send_keys('admin')
        driver.find_element(By.NAME, 'pass').send_keys('0000')
        driver.find_element(By.ID, 'buttonLogin').click()

    @allure.step("Step 2")
    def step2(self, driver):
        element = driver.find_element(By.ID, 'result')
        wait = WebDriverWait(driver, timeout=5)
        wait.until(lambda d: element.is_displayed())
        text = driver.find_element(By.ID, 'textarea').get_property('value')
        print("Get message: " + text)
        assert text == 'Authorization was successful', "Получено некорректное сообщение"

if __name__ == '__main__':
    pytest.main(["-s", "TestAuthorization.py"])
```

Открыть консоль и перейти в каталог с файлом автотеста TestAuthorization

Имя	Дата изменения	Тип	Размер
.pytest_cache	20.06.2025 10:46	Папка с файлами	
__pycache__	23.06.2025 16:52	Папка с файлами	
__init__.py	20.06.2025 10:14	JetBrains PyChar...	0 КБ
TestAuthorization.py	20.06.2025 15:00	JetBrains PyChar...	2 КБ

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5965]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests>
```

При запуске автотеста TestAuthorization указать путь к каталогу результатов теста в --alluredir аргументе командной строки:

```
python -m pytest -s -v TestAuthorization.py --alluredir allure-results
```

В результате будет выполнен автотест и создан каталог allure-results с данными для отчета.

Имя	Дата изменения	Тип	Размер
.pytest_cache	20.06.2025 10:46	Папка с файлами	
__pycache__	23.06.2025 16:52	Папка с файлами	
allure-results	23.06.2025 17:09	Папка с файлами	
__init__.py	20.06.2025 10:14	JetBrains PyChar...	0 КБ
TestAuthorization.py	20.06.2025 15:00	JetBrains PyChar...	2 КБ

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5965]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests>python -m pytest -s -v TestAuthorization.py --alluredir allure-results
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- C:\Program Files\Python312\python.exe
cachedir: .pytest_cache
rootdir: E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests
plugins: allure-pytest-2.14.3
collected 1 item

TestAuthorization.py::test_authorization
DevTools listening on ws://127.0.0.1:54642/devtools/browser/0b76e6d7-8dc1-4353-8880-b6f25fd03342
Get message: Authorization was successful
PASSED

===== 1 passed in 13.64s =====

E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests>
```

Запустите Allure командой, чтобы преобразовать результаты теста в HTML-отчет. Это автоматически откроет ваш браузер для просмотра отчета.

```
allure serve allure-results
```

Эта команда сформирует отчет и откроет его в браузере

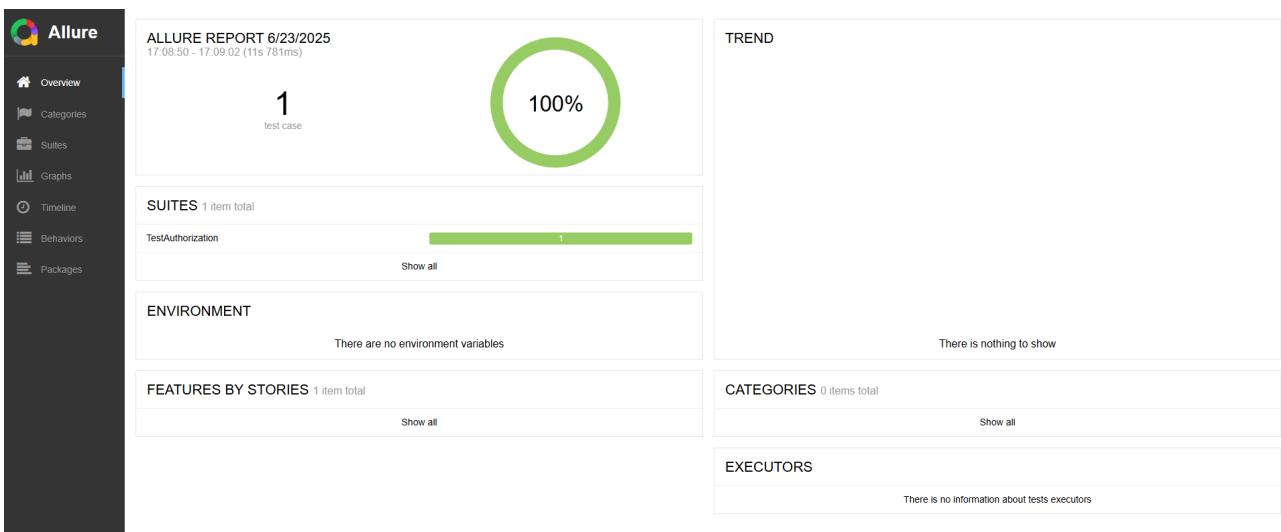
```
C:\Windows\System32\cmd.exe - allure serve allure-results
Microsoft Windows [Version 10.0.19045.5965]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests>python -m pytest -s -v TestAuthorization.py --alluredir allure-results
=====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- C:\Program Files\Python312\python.exe
cachedir: .pytest_cache
rootdir: E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests
plugins: allure-pytest-2.14.3
collected 1 item

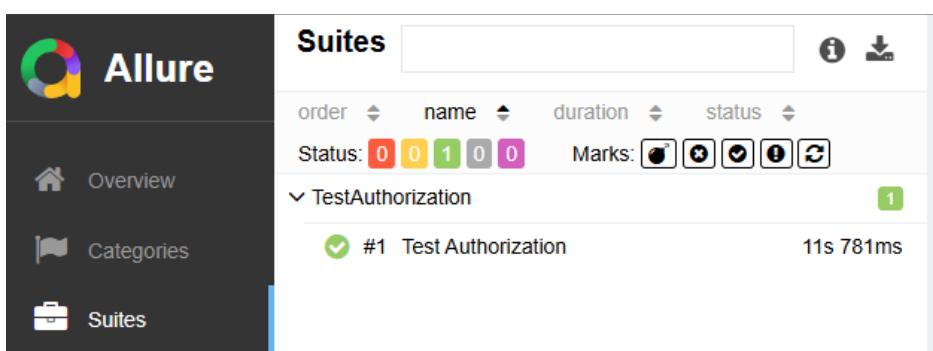
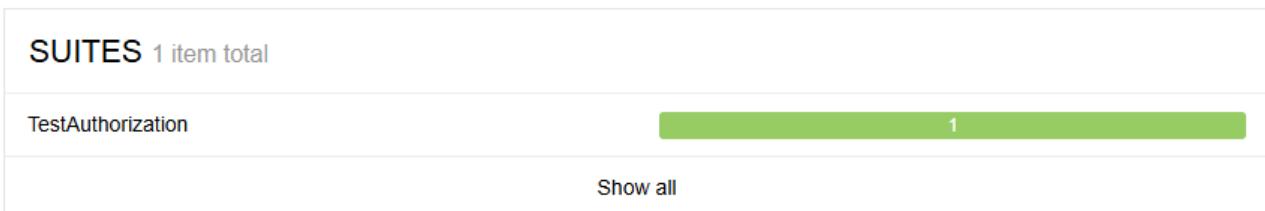
TestAuthorization.py::test_authorization
DevTools listening on ws://127.0.0.1:54642/devtools/browser/0b76e6d7-8dc1-4353-8880-b6f25fd03342
Get message: Authorization was successful
PASSED

===== 1 passed in 13.64s =====

E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests>allure serve allure-results
Generating report to temp directory...
Report successfully generated to C:\Users\Catfish\AppData\Local\Temp\7460258689107529984\allure-report
Starting web server...
2025-06-23 17:12:21.691:INFO::main: Logging initialized @2170ms to org.eclipse.jetty.util.log.StdErrLog
Server started at <http://192.168.132.1:54728/>. Press <Ctrl+C> to exit
```



Чтобы увидеть подробное описание выполненного автотеста нажмите на кнопку «Show all»



The screenshot shows the Allure Test Report interface. On the left is a sidebar with icons for Overview, Categories, Suites, Graphs, Timeline, Behaviors, and Packages. The main area is titled 'Suites' and shows a table with columns for order, name, duration, and status. A summary bar at the top indicates 0 failed, 1 passed, 0 skipped, and 0 pending tests. Below this is a tree view of suites, with 'TestAuthorization' expanded to show its single test case, which is highlighted in yellow and marked as passed.

Если автотест будет выполнен с ошибкой это будет отражено в отчете.

```
C:\Windows\System32\cmd.exe
steps = Steps()
steps.step1(driver)
>     steps.step2(driver)

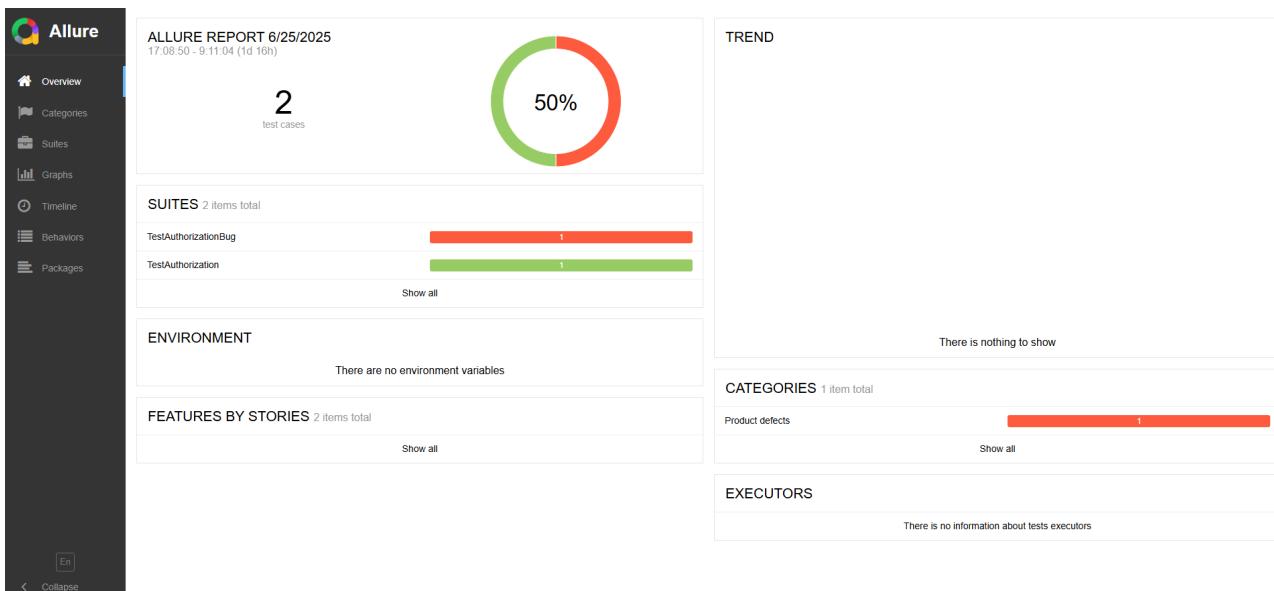
TestAuthorizationBug.py:24:
-----
self = <tests.TestAuthorizationBug.Steps object at 0x000001F6AEAA9C10>
driver = <selenium.webdriver.chrome.webdriver.WebDriver (session="d94b9856fc773f173b82dc4531ce7426")>

@allure.step("Step 2")
def step2(self, driver):
    element = driver.find_element(By.ID, 'result')
    wait = WebDriverWait(driver, timeout=5)
    wait.until(lambda d: element.is_displayed())
    text = driver.find_element(By.ID, 'textarea').get_property('value')
    print("Get message: " + text)
>         assert text == 'Authorization was successful', "Получено некорректное сообщение"
E             AssertionError: Получено некорректное сообщение
E             assert 'Invalid login or password' == 'Authorization was successful'
E
E                 - Authorization was successful
E                 + Invalid login or password

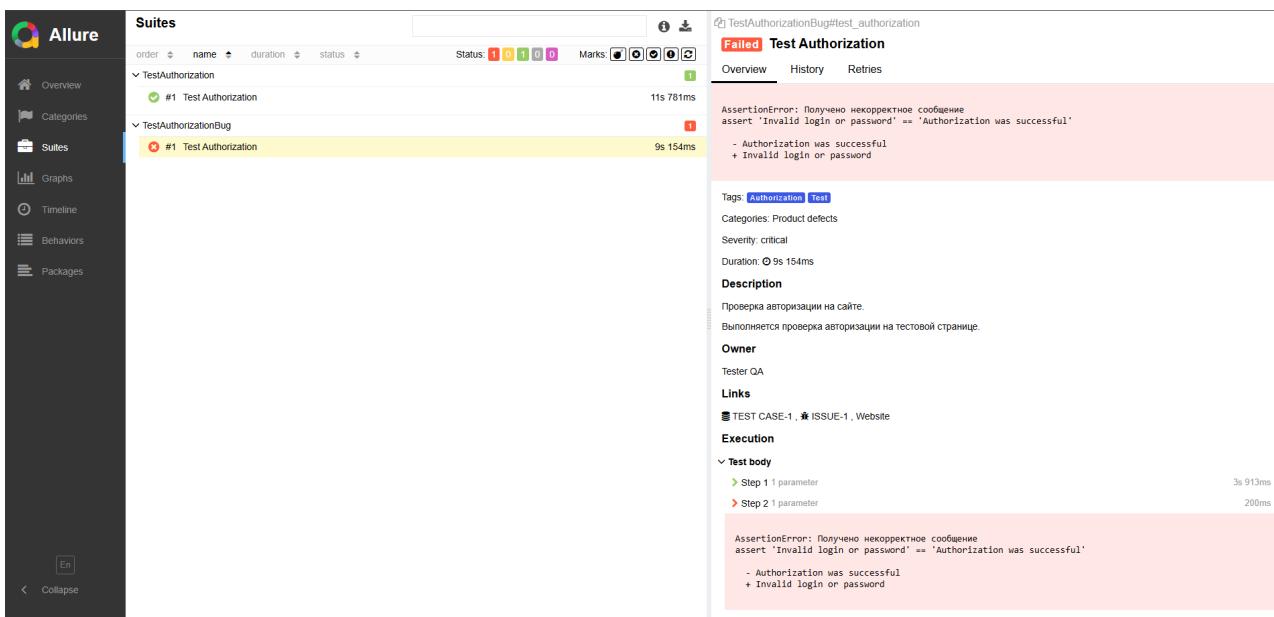
TestAuthorizationBug.py:45: AssertionError
===== short test summary info =====
FAILED TestAuthorizationBug.py::test_authorization - AssertionError: Получено некорректное сообщение
===== 1 failed in 13.95s =====
E:\Git\GITFLIC\autotesting\raports\allure_report\python\python-pytest-allure\tests>
```

Автотест с ошибкой выделен красным цветом.

SUITES 2 items total	
TestAuthorizationBug	<div style="width: 100%;">1</div>
TestAuthorization	<div style="width: 100%; background-color: #90EE90;">1</div>
Show all	



В описании неудачного автотеста можно увидеть причину ошибки



▼ Test body

- Step 1 1 parameter 3s 913ms
- Step 2 1 parameter 200ms

```
AssertionError: Получено некорректное сообщение
assert 'Invalid login or password' == 'Authorization was successful'

- Authorization was successful
+ Invalid login or password
```

Полезные ссылки:

- Быстрый старт Allure PyTest <https://pypi.org/project/allure-pytest/2.13.5/>
- Документация Allure PyTest <https://allurereport.org/docs/pytest/>

Отчет Report Portal

Скачать и

Jenkins

Скачать и

Docker

Скачать и

Нагрузочное тестирование с помощью Jmeter

Скачать и

Тестирование API с помощью Bruno

Чтобы протестировать API с помощью Bruno нужно выполнить следующие действия.
Скачать и установить [Wampserver](#) (сервер Apache, PHP, MySQL)
Создать на сервере тестовое API в папке \wamp64\www\api
файл: auth.php

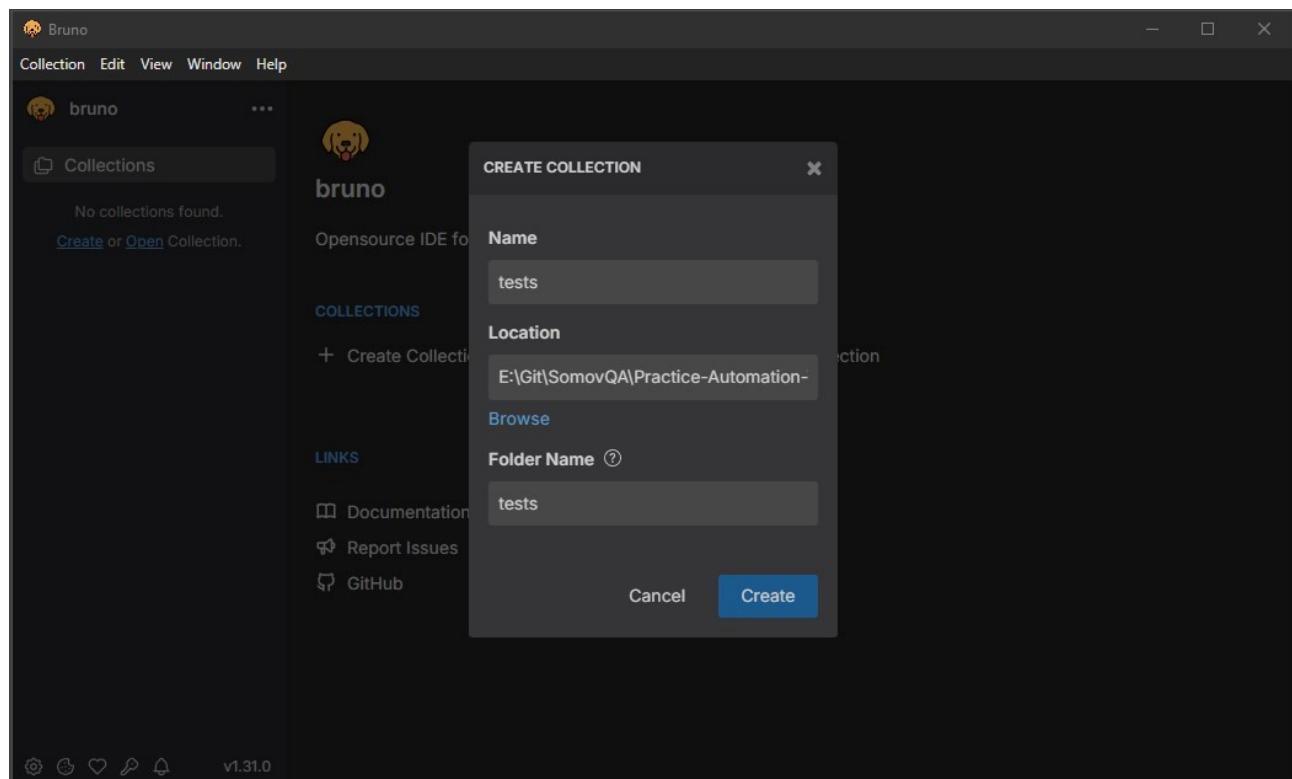
```
<?php

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
date_default_timezone_set("Europe/Moscow");

/*
    Пример обращения к этому API
    http://localhost/api/auth.php?name=admin&pass=0000
*/

if (isset($_GET["name"]) && isset($_GET["pass"]))
{
    if ($_GET["name"] == "admin" && $_GET["pass"] == "0000")
        print('{"status":"PASSED","message":"Авторизация прошла успешно"}');
    else
        print('{"status":"FAILED","message":"Некорректный логин или пароль"}');
}
elseif (isset($_POST["post_name"]) && isset($_POST["post_pass"]))
{
    if ($_POST["post_name"] == "admin" && $_POST["post_pass"] == "0000")
        print('{"status":"PASSED","message":"Авторизация прошла успешно"}');
    else
        print('{"status":"FAILED","message":"Некорректный логин или пароль"}');
}
else
{
    print('{"status":"ERROR","message":"Нет данных для авторизации"}');
}
?>
```

таким образом API будет по адресу <http://localhost/api/auth.php?name=admin&pass=0000>
Теперь можно скачать и установить [Bruno](#)
Запустить Bruno и создать коллекцию tests



В коллекции tests создать запрос Test01

The screenshot shows the Postman interface. On the left, there's a sidebar with options like 'New Request', 'New Folder', and 'New Script'. The main area is titled 'COLLECTIONS' with buttons for '+ Create Collection', 'Open Collection', and 'Import Collection'. A modal window titled 'NEW REQUEST' is open in the center. It has fields for 'Type' (set to 'HTTP'), 'Name' (set to 'Test01'), and 'URL' (set to 'GET http://localhost/api/auth.php'). At the bottom right of the modal are 'Cancel' and 'Create' buttons.

Выбрать один из режимов (safe - ограниченный, Developer - полный)

The screenshot shows the 'JAVASCRIPT SANDBOX' settings in Postman. It asks the user to choose a security level for JavaScript code execution. Two options are available: 'Safe Mode (BETA)' and 'Developer Mode (use only if you trust the collections authors)'. Below these options, a note states: '* SAFE mode has been introduced v1.25 onwards and is in beta. Please report any issues on github.' At the bottom right is a 'Save' button.

В новом запросе указать параметры

name: admin
pass: 0000

The screenshot shows the 'Params' tab in the request configuration. It lists two parameters: 'name' with value 'admin' and 'pass' with value '0000'. There are checkboxes next to each parameter entry, and icons for deleting the entries.

Name	Path	
name	admin	<input checked="" type="checkbox"/> trash
pass	0000	<input checked="" type="checkbox"/> trash

и заголовок

Content-type: application/json; charset=UTF-8

Params ²	Body [*]	Headers ¹	Auth	Vars	Script	Assert	Tests	Docs
Name	Value							
Content-type	application/json; charset=UTF-8	<input checked="" type="checkbox"/>						

нажать кнопку сохранить 

Выполните запрос

В результате сервер должен ответить: {"status": "PASSED", "message": "Авторизация прошла успешно"}

GET Test01 × +

GET http://localhost/api/auth.php?name=admin&pass=0000

Params² Body^{*} Headers¹ Auth Vars Script
Assert Tests Docs

Query

Name	Path
name	admin
pass	0000

+ Add Param

Path ⓘ

Name	Value
------	-------

Response Headers⁸ Timeline Tests
🔗 ↴ 200 OK 3ms 82B

```
1 {  
2   "status": "PASSED",  
3   "message": "Авторизация прошла успешно"  
4 }
```

Для того чтобы выполнить POST запрос параметры нужно передавать через Body

post_name: admin
post_pass: 0000

POST Test02 × +

POST http://localhost/api/auth.php

Params Body^{*} Headers Auth Vars Script^{*}
Assert Tests Docs Multipart Form

Key Value

Key	Value
post_name	admin
post_pass	0000

Response Headers⁸ Timeline Tests
🔗 ↴ 200 OK 3ms 82B

```
1 {  
2   "status": "PASSED",  
3   "message": "Авторизация прошла успешно"  
4 }
```

Простая проверка в которой определяется ответ сервера, если 200 значит успешно.

```
test("Проверить ответ сервера", function() {
  const data = res.getBody();
  expect(res.getStatus()).to.equal(200);
});
```

The screenshot shows the Postman interface with a test script and its results. The test script checks if the server returns a 200 status code. The results show 1 test passed and 0 failed, with a response time of 2ms and 82B.

Expr	Operator	Value	Action
res.status	equals	200	<input checked="" type="checkbox"/> ✖
res.body.message	equals	Авторизация прошла успешно	<input checked="" type="checkbox"/> ✖

Эту проверку можно выполнить без скрипта на вкладке Assert

The screenshot shows the Postman interface with assertions instead of a script. It checks if the status is 200 and if the message is 'Авторизация прошла успешно'. Both assertions pass.

Expr	Operator	Value	Action
res.status	equals	200	<input checked="" type="checkbox"/> ✖
res.body.message	equals	Авторизация прошла успешно	<input checked="" type="checkbox"/> ✖

Выполним проверку сообщения возвращаемого сервером

Создадим POST запрос и на вкладке Script пропишем отправляемые данные

```
req.setBody({
  "post_name": "admin",
  "post_pass": "0000"
});
```

The screenshot shows a POST request with a 'Pre Request' script. The script sets the body of the request to a JSON object with 'post_name' and 'post_pass' fields.

Expr	Operator	Value	Action
res.status	equals	200	<input checked="" type="checkbox"/> ✖
res.body.message	equals	Авторизация прошла успешно	<input checked="" type="checkbox"/> ✖

на вкладке Tests добавим проверку сообщения полученного от сервера

```
test("Проверить сообщения частично", function() {
  const data = res.getBody();
  expect(data.message).to.contains('успешно');
});

test("Проверить сообщения полностью", function() {
  const data = res.getBody();
  expect(data.message).to.equal('Авторизация прошла успешно');
});

test("Проверить тип сообщения строка", function() {
  const data = res.getBody();
  expect(data.message).to.be.a('string');
});
```

The screenshot shows the 'Tests' tab in Postman's interface. The code editor contains a Jest-style test script:

```
1 test("Проверить ответ сервера", function() {
2     expect(res.getStatus()).to.equal(200);
3 });
4
5 test("Проверить сообщения частично", function() {
6     const data = res.getBody();
7     expect(data.message).to.contains('успешно');
8 });
9
10 test("Проверить сообщения полностью", function() {
11     const data = res.getBody();
12     expect(data.message).to.equal('Авторизация прошла успешно')
13 };
14
15 test("Проверить тип сообщения строка", function() {
16     const data = res.getBody();
17     expect(data.message).to.be.a('string');
18 });
```

Выполним запрос



The screenshot shows the results of a test run for a POST request to 'http://localhost/api/auth.php'. The 'Tests' tab is selected.

Test	Status	Duration	Size
Проверить ответ сервера	200 OK	3ms	82B
Проверить сообщения частично	200 OK	3ms	82B
Проверить сообщения полностью	200 OK	3ms	82B
Проверить тип сообщения строка	200 OK	3ms	82B

Tests (4/4), Passed: 4, Failed: 0

- ✓ Проверить ответ сервера
- ✓ Проверить сообщения частично
- ✓ Проверить сообщения полностью
- ✓ Проверить тип сообщения строка

Assertions (0/0), Passed: 0, Failed: 0

В результате будет проверено сообщение частично и полностью, а так же проверен тип поля.

Запуск тестов из командной строки

Скачайте и установите [NodeJS](#)

Откройте консоль и выполните установку

```
npm install -g @usebruno/cli
```

```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>npm install -g @usebruno/cli
npm warn deprecated har-validator@5.1.5: this library is no longer supported
added 284 packages in 1m

33 packages are looking for funding
  run `npm fund` for details

C:\Users\Catfish>
```

Перейдите в каталог, где находится коллекция запросов и выполните следующую команду:

```
bru run test02.bru
```

результат выполнения будет отражен в консоли

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests>dir
Том в устройстве E не имеет метки.
Серийный номер тома: 5A79-A271

Содержимое папки E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests

03.10.2024 13:18    <DIR>      .
03.10.2024 13:18    <DIR>      ..
03.10.2024 10:40            113 bruno.json
03.10.2024 13:25            521 Test01.bru
04.10.2024 09:37            1 045 Test02.bru
                           3 файлов        1 679 байт
                           2 папок   158 880 210 944 байт свободно

E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests>bru run test02.bru
Running Request

test02 (200 OK) - 117 ms
headers: [object Object]
method: POST
url: http://localhost/api/auth.php
  Проверить ответ сервера
  Проверить сообщения частично
  Проверить сообщения полностью
  Проверить тип сообщения строка

Requests:  1 passed, 1 total
Tests:     4 passed, 4 total
Assertions: 0 passed, 0 total
Ran all requests - 117 ms

Requests:  1 passed, 1 total
Tests:     4 passed, 4 total
Assertions: 0 passed, 0 total

E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests>
```

Другие команды

- Чтобы выполнить все запросы в папке, используйте:

```
bru run folder
```
- Если вам необходимо использовать определенную среду, вы можете передать ее с помощью параметра --env:

```
bru run folder --env Local
```
- Вы можете передавать переменные среды непосредственно в свою коллекцию с помощью параметра --env-var:

```
bru run folder --env Local --env-var JWT_TOKEN=1234
```
- Чтобы сохранить результаты тестов API в файл, используйте опцию --output:

```
bru run folder --output results.json
```
- Для формирования отчета в формате JSON используйте --reporter-json опцию:

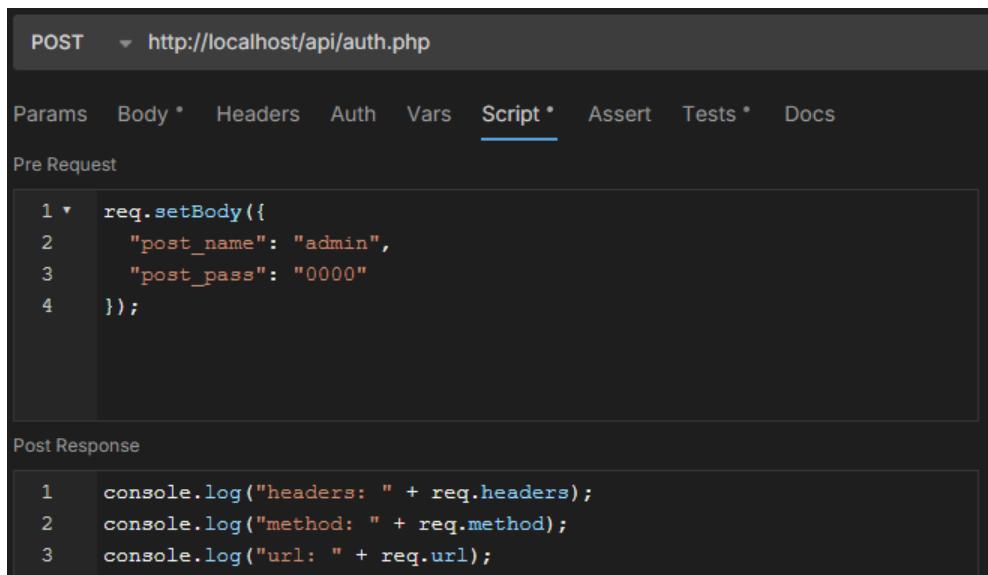
```
bru run request.bru --reporter-json results.json
```

- Чтобы создать отчет в формате JUnit, используйте опцию --reporter-junit:
`bru run request.bru --reporter-junit results.xml`
- Чтобы создать отчет в формате HTML, понятный человеку, используйте опцию --reporter-html:
`bru run request.bru --reporter-html results.html`
- Одновременный запуск нескольких репортеров
`bru run request.bru --reporter-json results.json --reporter-junit results.xml --reporter-html results.html`

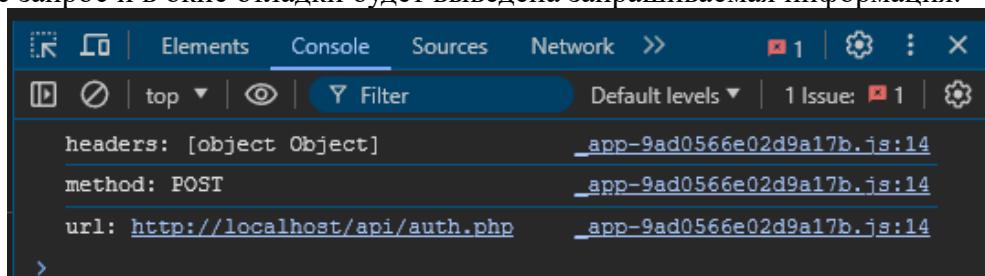
12. Отладка в окне Toggle Developer Tools

На вкладке Script добавьте вывод информации в консоль

```
console.log("headers: " + req.headers);
console.log("method: " + req.method);
console.log("url: " + req.url);
```



Нажмите меню "View" > "Toggle Developer Tools" чтобы открыть окно отладки.
Выполните запрос и в окне отладки будет выведена запрашиваемая информация.



Полезные ссылки:

- Официальная страница Bruno
<https://docs.usebruno.com/>
- Официальная документация API Testing
<https://docs.usebruno.com/testing/introduction>
- Официальная документация Scripting
<https://docs.usebruno.com/scripting/getting-started>
- Официальная документация CLI
<https://docs.usebruno.com/bru-cli/overview>

Локаторы XPATH и сравнение с CSS

Язык запросов XPath к элементам XML-документа

child::	элементы потомки
descendant::	полное множество элементов потомков
descendant-or-self::	полное множество потомков и текущий элемент
ancestor::	элементы предки
ancestor-or-self::	элементы предки и текущий элемент
parent::	элемент предок на один уровень назад
self::	текущий элемент
following::	элементы ниже текущего элемента
following-sibling::	братьеские элементы того же уровня
preceding::	элементы выше текущего уровня
preceding-sibling::	братьеские элементы предшествующие текущему
attribute::	атрибуты текущего элемента
namespace::	элементы относительно пространства имен
*	все элементы

Функции над множествами узлов (n - это node-set; o - это object)

node()	возвращает сам узел
text()	возвращает текст узла
current()	возвращает множество из текущего элемента
position()	возвращает позицию элемента в множестве элементов
last()	возвращает номер последнего элемента в множестве
count(n)	возвращает кол-во элементов в узле
name(n)	возвращает полное имя первого тега в множестве
namespace-url(n)	возвращает ссылку на URL
localname(n)	возвращает имя первого тега без пространства имен
id(o)	находит элемент с уникальным ID

Строковые функции

string(obj)	возвращает текстовое содержимое элемента
concat(str, str, str*)	соединяет строки
length(str)	возвращает длину строки
contains(str, str)	возвращает true если первая строка содержит вторую
substring(str, num, num?)	возвращает вырезанную строку
substring-before(str, str)	если найдена вторая строка в первой возвращает строку до

	первого вхождения второй строки
substring-after(str, str)	если найдена вторая строка в первой возвращает строку после первого вхождения второй строки
starts-with(str, str)	возвращает true если вторая строка входит в начало первой (иначе false)
end-with(str, str)	возвращает true если вторая строка входит в конец первой (иначе false)
normalize-space(str)	удаляем лишние и повторные пробелы
translate(str, str, str)	заменяет символы в строке

Логические функции и операторы

and	логическое И
or	логическое ИЛИ
=	логическое равно
<	логическое меньше
>	логическое больше
<=	логическое меньше или равно
>=	логическое больше или равно
boolean(obj)	приводит объект к логическому типу
true()	возвращает истину
false()	возвращает ложь
not(boolean)	отрицание, возвращает истину если аргумент ложь

Числовые функции и операторы

+; -; *;	сложение, вычитание, умножение
div	деление
mod	остаток от деления
number(obj)	переводит объект в число
sum(node)	возвращает сумму множества
floor(number)	возвращает наибольшее целое число
ceiling(number)	возвращает наименьшее целое число
round(number)	округление

Системные функции

document(obj, node)	возвращает документ
format-number(num,str,str)	формат числа
generate-id(node)	уникальный идентификатор
key(str, obj)	возвращает множество с указанным ключом

unparsed-entity-uri(str)	возвращает не проанализированный URL
element-available(str)	проверяет доступен ли элемент
function-available(str)	проверяет доступна ли функция
system-property(str)	возвращает системные параметры
lang(str)	возвращает true если у текущего тега есть

Прочие обозначения

*	любое имя
@name	имя переменной или параметра (@id, @class)
[]	дополнительные условия выбора
{ }	если применяется внутри тега другого языка
/	определяет уровень дерева
	объединяет результат

Примеры:

```
//div[@id='example_id']//div/a
//div[@class='example_class']//div/a
//input[@id='user']/following-sibling::input[4]
//input[@name='login' and @type='submit']
//*[contains(text(), 'login')]
//a[contains(@href, 'google.com')]
//label[contains(text(), 'интернет')]/parent::div
```

Сравнение запросов CSS и Xpath

CSS	XPATH
div > a	//div/a
div a	//div//a
#user_id	//div[@id='user_id']
.user_class	//div[@class='user_class']
#login + input	//div[@id='login']/following-sibling::input[1]
input[name='user']	//input[@name='user']
input[id='btn'][type='submit']	//input[@id='btn' and @type='submit']

Использование нескольких имен классов

```
<div class="value test"></div>
<div class="value test "></div>
<div class="first value test last"></div>
<div class="test value"></div>
```

div[class='value test ']	//div[@class='value test ']
div[class*='value test']	//div[contains(@class, 'value') and contains(@class, 'test')]
div.value.test	

Дочерние элементы

```
<ul id="list">  
  <li>value1</li>  
  <li>value2/>  
</ul>
```

#list li:nth-of-type(4) #list li:nth-child(4) #list *:nth-child(4)	//li[1]/parent::ul
--	--------------------

Частичное совпадение строк

```
a[id ^= 'id_prefix_']  
a[id $= '_id_sufix']  
a[id *= 'id_pattern']
```

Соответствие по внутреннему тексту

a:contains('log out')	//a[contains(text(), 'log out')
-----------------------	---------------------------------

...